12

AD-A141 648

RADC-TR-83-308
Final Technical Report
January 1984

ADAPTIVE MODELING AND REAL-TIME SIMULATION

SRI International

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

This effort was funded totally by the Laboratory Directors' Fund

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

84 05 29 030

RADC-TR-83-308 has been reviewed and is approved for publication.

APPROVED: [illegible]

NORTHRUP FOWLER III
Project Engineer

APPROVED: [illegible]

RAYMOND P. URTZ, JR.
Acting Technical Director
Command and Control Division

FOR THE COMMANDER: [illegible]

JOHN A. RITZ
Acting Chief, Plans Office

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-308 / 2. GOVT ACCESSION NO. AD-A141648	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADAPTIVE MODELING AND REAL-TIME SIMULATION	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Jun 81 - Jun 83
	6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) William M. Tyson	8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0218
9. PERFORMING ORGANIZATION NAME AND ADDRESS SRI International 333 Ravenswood Ave Menlo Park CA 94025-3493	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61101F LDFP09C1
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COES) Griffiss AFB NY 13441	12. REPORT DATE January 1984
	13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Northrup Fowler III, (COES)
This effort was funded totally by the Laboratory Directors' Fund

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial Intelligence	Truth Maintenance
Planning	Resolution
Modeling	World Models
Simulation	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is the final report covering progress on a two year research effort towards the development of basic technology for adaptive modeling and real-time computer simulation to support decision-making in a number of critical planning situations that arise during the execution of tactical air missions. Both tactical and defensive planning must be done quickly--the side that is faster and better prepared will have the advantage. Still, plans must be accurate. Planning too quickly may cause important informa-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

tion to be overlooked--information that may affect whether the plan will achieve its goal. Computers should be able to support decision-making and planning, but currently, for a number of reasons, they do not approach their potential use in this field. One major reason is that understanding of planning and modeling of real world situations are inadequate. These adequacies involve:

- World models;
- A model of time;
- Understanding of inaccurate information,
- Propagating the effects of information and retracing (backtracking) that propagation if necessary, and
- Processing speed, especially as regards deduction and simulation.

In this report ~~we have~~ reviewed these difficulties and developed approaches to solving them. these

An annotated bibliography of all the major articles pertaining to the subject domain is included in an appendix.

CONTENTS

I INTRODUCTION AND STATEMENT OF THE PROBLEM

A. Introduction

This is the final report covering progress on a two year research effort towards the development of basic technology for adaptive modeling and real-time computer simulation to support decision-making in a number of critical planning situations that arise during the execution of tactical air missions. The control of such missions has become increasingly difficult and complex. Several factors contribute to this complexity. For one, tactical air missions are carried out in highly dynamic, hostile environments where significant changes can occur in a conflict situation within relatively short periods of time. For another, the rate of information generated during combat has greatly increased as a result of improved sensing and communication capabilities. Finally, situation changes are often not accurately projected, because planning and decision-making are carried out on the basis of incomplete, uncertain knowledge as to future resource availability and enemy deployment. These factors make effective control during execution of tactical air missions both difficult and critical.

In the next section we present a brief overview of the current structure of the Tactical Air Control System in the U.S. Air Force. This will serve to introduce our general problem of interest while placing it in the appropriate context. Following this is a discussion of the capabilities required of a system that could provide real-time support for replanning of previously scheduled activity. A hierarchical capability taxonomy is described and a scenario for interactive use of such a system is presented. These constitute a statement of goals for this technology. We then discuss how we intend to realize these goals, specifying a design for our system in terms of an abstract, semantic model of our problem domain and a functional description of the system. The abstract model represents a data dictionary of information entities to be manipulated within the system. The functional description includes a functional decomposition of system tasks and a data flow analysis in terms of entities described in the abstract model. We conclude this design plan with an example that illustrates both the representation of mission control problems within the abstract model and the application of the proposed system to these problems.

B. The Context and Problems

The real-time management of tactical air forces takes place within the command and control hierarchy of the Tactical Air Control System (TACS). Our model of TACS is based on a report prepared for the U.S. Air Force by the RAND Corporation [26] and a survey article on techniques and problems of force management decision-making in a tactical-air-force context [68].

At the top of the TACS hierarchy is the Tactical Air Force Headquarters (TAFHQ), consisting of the commander (COMTAF) and staff. TAFHQ is responsible for the overall direction and long-range planning of air war operations. This includes the specification of air strategy based on descriptions of attractive targets, locations and numbers of available resources, and gross (percentage) apportionment of those resources to target types. This information is provided to the central element of the TACS hierarchy, the Tactical Air Control Center (TACC).

The TACC has primary responsibility for the day-to-day planning and control of tactical air activity. The TACC transforms strategic guidelines from TAFHQ into specific tactical air missions by selecting actual targets and allocating available forces on a daily basis. It must coordinate its designations of tactical air missions—air interdiction and offensive counter air missions—with subordinate elements that bear responsibility for defensive, air-lift, and air-support missions. The Wing Operations Center (WOC) must implement the missions provided by the TACC—completing detailed (flight) plans, assigning aircraft and crews, and launching missions as scheduled. Each WOC is associated with, and located at, an air base; a TACC may control several WOCs. Finally, at the bottom of the hierarchy, assigned forces execute the planned missions.

Our interests focus on activities within the TACC. Not only must the TACC generate plans for an upcoming day's activities, but it must monitor the execution of those plans, modifying them as necessary to reflect the ongoing conflict situation. To carry out these two primary functions, the TACC is divided into two sections: Combat Plans and Combat Operations. Combat Plans is responsible for "tomorrow's war," providing an Air Tasking Order (ATO, or 'frag order') to Combat Operations prior to each day's action. Combat Operations monitors the ongoing air war, deciding whether to continue with or modify to an extant ATO. Combat Operations conducts "today's war." In monitoring current activity, Combat Operations receives inputs of various kinds. These include mission reports that disclose launch, inflight, target outcome, and landing information, intelligence reports that indicate enemy activity as well as weather conditions, resource availability reports, and requirement reports for other types of missions as issued by the CRC (defensive air), ALCC (airlift), and ASOC (air support). Combat Operations must determine the significance of the new information contained in received reports, evaluating it relative to expectations based upon the currently accepted ATO. It must then decide whether to adhere to the plans represented in the ATO—or how to modify them so as to best carry out the strategy indicated by TAFHQ within the constraints posed by a rapidly evolving combat situation.

Combat Operations is that part of the TACS hierarchy that is concerned primarily with real-time resource management. A Combat Operations decision to modify an extant ATO may take one of the following forms: a redirection of inflight aircraft to a new target, an assignment of ready (on alert) aircraft to a target, a reallocation of resources to new targets, or an allocation of previously unused resources to a target. Combat Operations must decide whether to post, reschedule, or cancel missions in light of newly received reports.

The activity that takes place in Combat Operations exemplifies the problem confronted by all systems that attempt to carry out plans in real-world contexts: the need for execution monitoring and real-time replanning of previously scheduled, planned activity. We now turn our attention to the capabilities required of a system that can provide support in carrying out this activity.

II SYSTEM CAPABILITIES

In this section we describe the capabilities needed in a system that can provide support for monitoring, control, and replanning of previously planned activity. As such, they represent capabilities to be realized by the system we are designing. Here we only describe system goals—what it should do, not how it might do it; aspects of system design—the hows—are discussed in the following sections.

Within a capability taxonomy for our system, three general classes of necessary capabilities can be distinguished:

- Provide data base facilities for current plans.
- Determine import of new information with respect to current plans.
- Evaluate impact of proposed modification of current plans.

We consider each of these classes in the following discussion. We conclude by briefly describing an anticipated scenario for the interactive use of our proposed system.

A. Providing a Current-Plans Data Base

If a system is to interact with its users regarding some content domain, it must be able to perform basic data base functions with respect to that domain In other words, it must be able to represent, store, access, and update information about its domain of discourse. For our application domain, a satisfactory system must be able to represent situations in the task environment, goals and plans of the active agent(s), as well as the relations that exist among these entities. The system must provide the facilities for adding, deleting, and otherwise modifying entities and relations of a data base that represents planned activity. It must provide mechanisms for selectively accessing elements of this extant data base. Finally, it must provide a serviceable interface with the user and other subsystems, affording easy access to these data base facilities. Increasing amounts of information are being made available for decision-making in all domains as a result of improved intelligence and communication systems. If better decisions are to follow, the information must be quickly and naturally accessible.

Of particular importance in our domain of discourse will be the representation of temporal information. Planned activity is to be executed over some period of time.

Aspects of planned activity have associated schedule information. This implies the need to accommodate multiple instances of task environment situations in any extant data base representing planned activity; each instance must have associated with it appropriate temporal information. Explicit or assumed time specifications must be interposed when data base information is being accessed with regard to situations occurring in the task environment. Conversely, such access to information must be made relative to explicit or assumed time specifications.

B. Determining the Significance of New Information

The meaning of information acquired during the execution of planned activity can be determined only relative to that planned activity. A satisfactory system must be able to evaluate whether new information is consistent with expectations that are implied or assumed by the planned activity. If it is not consistent, the system must be able to characterize possible conflicts and indicate their potential significance with respect to planned activity.

New information may clash with expectations in several ways. The system we propose will be capable of discerning several types of conflicts, including the following:

- A resource is not (will not be) available when needed.
- A required condition in the task environment is (will be) violated.
- A goal is (will be) satisfied without executing its associated plan.
- An aspect of planned activity is not executed as scheduled.
- An aspect of planned activity is not executed successfully.

These conflict types will be determined relative to a past, the present, or a future time. The third conflict type, while not agreeing with expectations, represents a fortuitous turn of events. The system should be able to recognize positive circumstances as well as those with negative import.

In order to indicate the apparent and potential significance of newly acquired information, a satisfactory system must be able to propagate the effects of the conflict to relevant aspects of planned activity. The process of conflict propagation requires two more basic capabilities: (1) limited deductive reasoning to determine goal- and plan-related implications, and (2) selective discrete-event simulation to determine time-related, plan interaction implications.

For example, suppose new information indicates that a scheduled segment of planned activity has been delayed. By deductive reasoning about relations of the plan that includes that segment, it may be determined that subsequent portions of the plan cannot be executed as scheduled, since they depend upon prior completion of the delayed segment. However, such reasoning may also indicate the presence of sufficient leeway in that plan's schedule to accommodate the delay. On the other hand, by selectively simulating effects of the reported delay on resource availability, it may be determined that even though no problem arises within the plan of the delayed segment, portions in other plans cannot be executed when scheduled because of resource use conflicts. Subsequent deductive reasoning about relations of those plans may or may not indicate the leeway now required for rescheduling of those portions.

We see deduction and simulation as complementary capabilities in our proposed system for providing decision support in the control of planned activity. They play fundamental roles not only in determining the significance of information acquired during execution, but also in evaluating the effect of proposed modifications of planned activity.

C. Evaluating the Effect of Proposed Modifications

Effective control of planned activity can be defined as the ability to modify plans commensurately in response to an evolving execution context. As discussed above, this depends upon understanding the significance of newly acquired information about that context. Likewise, the degree to which proposed modifications may affect presently planned activity must be determined. Several types of plan modifications are possible: the cancellation, delay, or other alteration of some aspect of planned activity, the addition of new aspects, and the reallocation of resources among aspects.

The modification of planned activity results in the creation of new information and so shares a basic requirement with the acquisition of information: conflicts with current expectations must be detected and evaluated. Plan modifications can be viewed as new information; new information can be viewed as execution-time modifications of plans. Conflict interactions may have positive or negative import, as noted above. The types of conflicts that arise from modifications of planned activity include those that result from the receipt of new information. Rescheduling of an aspect of planned activity or the addition of a new aspect can lead to resource use conflicts, violation of the prerequisites of other aspects, or denial of standard operating procedures (e.g., a certain number of resource units must be kept in reserve, sufficient time must be allowed for possible error recovery). Positive interactions can occur; a newly added aspect may subsume an existing aspect, while satisfying new goals as well.

As in determining the significance of new information, we propose that deduction and simulation play fundamental, complementary roles in evaluating the effects of proposed modifications of planned activity. Simulation can be used to incorporate a plan change into a temporal sequence of expected situations in the task environment. Deduction can determine inconsistencies and propagate these according to current plan relations (irrespective of time). Simulation can propagate inconsistencies to appropriate points throughout the situation sequence.

Before turning our attention to an anticipated scenario of system use, the question of system responsiveness must be addressed. If a system is to provide adequate support for the decision-making required in the effective control of planned activity, it must perform within real-time constraints imposed by the context. What can be considered real-time, however, differs according to the type of context and with the level of control within a given context. In proposing a system based upon data base, deduction, and simulation components, we do not mean to give the impression of ignoring issues of system responsiveness. We envisage application of the proposed system at a level and in a context requiring control cycles (i.e., feedback loops) on the time scale of minutes to an hour. To satisfy these real-time constraints, computational complexity must be controlled through appropriate data base representation, limited deduction, and selective simulation.

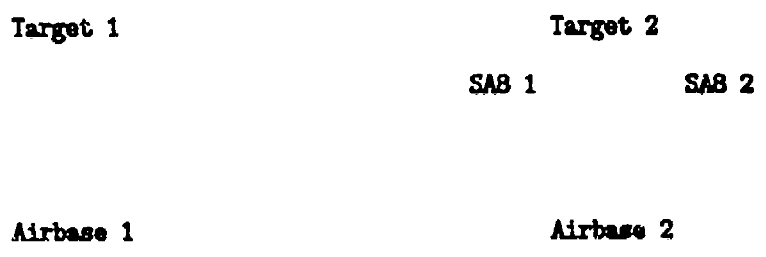
D. Scenario of System Use

We conclude this section with a brief description of how our system might be used in the control of planned activity. These uses reflect the basic capabilities of the system as described above. Appendix A contains a partial model for the original plans of this scenario.

We assume that a data base representing activity as currently planned has been previously established, as is the case in Combat Operations. We note that the system could play a supportive role in establishing a set of mutually consistent plans prior to execution time. Let us assume that execution of planned activity has begun and that a report of new information has been received. The user (e.g., commander, aide) submits this information to the system. The system ascertains the significance of the information, as described above, and reports its finding to the user. If all is well or if conflicts can be automatically accommodated (as in the case of a delayed task where adequate leeway in schedules exists), the interaction sequence is completed and the user awaits new information. However, any conflicts requiring, or opportunities suggesting, modifications of planned activity may be noted by the system. The user could then query the system as to resource availability or environmental conditions expected at that or some future time. The user could also inquire as to the reasons for (goals of) certain aspects of activity. This interaction might lead to a proposed modification, which is then submitted to the system. The system must incorporate this change into a new version of its data base and evaluate any consequent effects upon other current plans. This could lead to a revision of the proposal or even further proposals to overcome detected, undesirable side effects.

The following scenario should give an idea as to how we expect the designed system to work. In an actual system all the details of the plans would be included. Some preconditions for attacking a target (such as that the target must be relatively unprotected, that the aircraft be available for the duration of the attack, and that the aircraft be properly armed beforehand) will be mentioned as needed here. While, in a working system, of course all such preconditions would be included. We shall also deal with time loosely in this scenario, whereas it would be handled much more precisely in the designed system. The schematic layout is intended merely to provide some intuition as to what the world would look like. The system would actually have the exact positions of all the important entities. Other details, such as what aircraft and armaments would be used and the flying speeds of the aircraft, would be required by the system.

Schematic Layout:



Initial Plans:

(1) Destroy Target 1, which is relatively unprotected (presumably because of an earlier attack).

(2) Destroy Target 2. This involves an initial attack to knock out the SA8 sites. Airbase 2 will provide the aircraft required.

Problem:

Two hours before the launch of the attack against Target 1, reports indicate that some SA8 missiles will be in position within the hour to defend Target 1.

The system should realize that this presents problems for the mission to destroy Target 1. One precondition for that mission, that Target 1 be relatively unprotected at the time of attack, had been presumed true but is now in doubt. After comparing the expected arrival of reinforcements with the scheduled arrival time of the mission at the target, the system will determine that the precondition is indeed false.

The system would have several possible lines of action to attain the precondition of the target's being relatively unprotected. First, the defense may be avoided by either flying around it or by arriving before the defense is in place. Second, the defense may be destroyed during either the current mission or an earlier. Third, the defense may be neutralized by jamming or otherwise interfering with its operation. Fourth, the defense may be flown through at the risk of losing aircraft. Finally, the mission may be canceled. These options are presented to the operator as prospective courses of action.

Suppose the operator chooses to try to advance the arrival time of the mission at the target. The system analyzes this and finds that, since the remaining preparation time required for the aircraft combined with the flight time exceeds one hour, this plan modification is not sufficient to restore the necessary precondition.

After being informed of this, the operator checks for any jamming aircraft available to accompany the mission, but none are available.

The operator now chooses to destroy the SA8s by another mission. He indicates to use aircraft from Airbase 1. The system replies that this is impossible because none of the aircraft available have the required armament. Alternative suggestions are to divert some aircraft from another mission or to send some from another base.

The operator then decides to send some from Airbase 2, but the system again reports that there are no unassigned aircraft available with the required armament. The operator then tries to reassign the aircraft that have been assigned to destroy SA8 Site 1 (and that possess the proper armament to attack Target 1). The system realizes that this interferes with the second plan, i.e., to destroy Target 2, but looks to see whether a rescheduling is possible.

There is a time limit within which Target 2 must be attacked—no later than 8 hours after the destruction of the initial SA8 site. If another mission were immediately set up to destroy SA8 Site 1, the mission to attack Target 2 would have to be delayed but could still fit within the allotted time frame.

The operator verifies all these actions and orders are given to reassign the mission against SA8 Site 1 to attack the SA8 defending Target 1. Another mission is launched against SA8 Site 1 and the mission to attack Target 2 is rescheduled.

Features:

Upon being notified of a change in the situation, the system determines that this invalidates a presumed precondition. The resulting problem is serious enough so that the system does not try to determine what course of action to take on its own. Instead, it presents some set of potential actions for the human operator to consider.

The operator responds with some particular action, but the system analyzes it and finds that it does not remedy the problem. Again the operator is informed of this. This happens several times.

Eventually a suitable plan is devised. The system verifies that there is a way of satisfying all the preconditions and attaining all the goals. This may involve some minor rescheduling of existing plans. The final plans are then displayed for the operator's approval.

In analyzing the plans, the system must know both the plans and the present status of each mission. In addition, to predict future status, the system must know the relative positions of all the entities (bases, targets, aircraft, etc.) and their velocities. Of course, it must also know the resources available for use.

For the purposes of this project, the proposed system will not perform the replanning automatically. However, in some cases, it will be able to suggest straightforward options. Complete replanning, just like the original planning stage, would appear to involve complex knowledge of the tactics of a situation and an appreciation of goal evaluations based upon more global, strategic criteria. We do not perceive the knowledge-engineering effort required to realize effective, automatic replanning as being within the intended scope of this project.

III TEMPORAL LOGIC

One of the most active areas in current planning research concerns the temporal nature of plans and the predicates upon which they are based Planning systems for time-critical events in real-world applications will not be possible without a proper treatment of time. A system for replanning during execution will depend even more critically upon the system's temporal proficiency. Our system will use temporal predicates for the statement of STATE AXIOMS.

A. Inadequacy of State-Based Systems

In mathematics, predicates are simply true or false. If a theorem prover detects that a predicate is true and then later detects that it is false, it has found a contradiction from which it could prove anything. In planning, as soon as any action of a plan occurs, the truth of predicates[*] in the world model may change. A system for planning or replanning must be able to follow the altered world model without deriving contradictions.

During the planning process, predicates may change values as the planner asserts that an action ("operator" in STRIPS [22]) is added to the plan. Thus, when an operator *TURN-LIGHT-OFF* is added to the plan, the value of the *LIGHT-ON* predicate would be (possibly) changed in the world after the action of turning the light off has been performed.

Since the state of the world changed with each action, the early planning systems were state-based. A state was defined by a set of predicates true in that state. An action transformed the world from one state to another. Each state had its own set of predicates. The state of the world before the *TURN-LIGHT-OFF* might have the LIGHT-ON predicate as true, while the state after the action would have it as false. No contradictions existed within any state. Time was modeled as the sequence of states.

State-based systems are computationally attractive. Each state can be represented as a collection of predicates. All computations to determine state transformations need consider only the predicates within the present state. When an action is proposed, the

[*] We shall normally call a predicate *P* true if *P* is asserted and false if *Not P* is asserted.

state resulting from that operation can be quickly calculated and kept separate from the earlier state. Should that action be rejected for later reasons, backtracking to the earlier state is easy. Many simple actions can be modeled by a state-based system.

However, when the actions become more complex, state-based systems are no longer appropriate. The critical shortcoming of these systems is the simplistic modeling of time. Continuous changes (e.g., position of a moving object) cannot be modeled by a state-based system. Coordination of planned events with external events is more natural if the planning system incorporates time in its model.

Some state-based systems take a first step towards the embodying of time. Instead of keeping the predicates in each state separate, every predicate takes an extra argument indicating in which state to evaluate it. Thus, $LIGHT\text{-}ON(s_0)$ may be true while $LIGHT\text{-}ON(s_1)$ would be false if s_0 were the state before performing *TURN-LIGHT-OFF* and s_1 is the state afterwards. From this, it seems only a short step to describing LIGHT-ON(t) to be a function of time. Operators could then be described as occurring at a particular time (rather than in a state). Thus if *TURN-LIGHT-OFF* occurs at time t_0, $LIGHT\text{-}ON(t)$ is false for $t \geq t_0$.

Since a state was a set of predicates true at some point in time, we shall represent that time point as $TIME(s_0)$ for state s_0. We could then express a state as $STATE(t)$, meaning the set of all predicates true at time t. Thus

$$s_0 = STATE(TIME(s_0))$$

is a tautology.

B. Temporal Predicates

We have chosen a slightly different notation than the usual state-based notation to indicate the temporal dependence of predicates. We shall employ statements in a temporal logic similar to that described in [1]. This logic is a typed, first-order predicate calculus, in which terms are either condition statements, points in time, or intervals of time. We shall represent the interval from time t_1 to time t_2 by the notation $[t_1,t_2]$. The time interval is assumed to be closed at t_1 and open at t_2, capturing the notion of "from t_1 up to (but not including) t_2." By this assumption, t_2 must be greater than t_1, thus guaranteeing that all time intervals represent some finite, nonempty interval of time.

Three basic predicates are *HOLDSAT*, *HOLDSOVER*, and *HOLDSDURING*. Instead of $LIGHT\text{-}ON(t)$, we use $HOLDSAT(LIGHT\text{-}ON,\ t)$. The statement

HOLDSAT(*cs*, t) represents that predicate *cs* is true at time t. *HOLDSAT*(*cs*, t) is true if *cs* is true with respect to *STATE*(t), where *STATE*(t) is the complete state accepted as true at time t. *HOLDSOVER*(*cs*, $[t_1,t_2]$) represents the same for *cs* throughout time interval $[t_1,t_2]$.* *HOLDSOVER*(*cs*, $[t_1,t_2]$) is true if *cs* is true with respect to *STATE*(t) for all t, $t_1 \leq t < t_2$. *HOLDSDURING*(*cs*, $[t_1,t_2]$) represents the existence of the satisfaction of *cs* at some point within the interval $[t_1,t_2]$. *HOLDSDURING*(*cs*, $[t_1,t_2]$) is true if there exists t, $t_1 \leq t < t_2$, such that *cs* is true with respect to *STATE*(t). Time is considered to be ordered and dense; as such, we model it by the real numbers. Other predicates express relationships between time points and/or intervals (e.g., *BEFORE*, *DURING*). Since this is a predicate calculus, quantifiers and logical connectives are available. Allen [1] presents a basic set of axioms for the resultant temporal logic.

There are two reasons for using this temporal logic rather than simply adding time as an extra argument to each predicate. First, within a particular state, s_0, time is constant and can be ignored. The *HOLDSAT*(*cs*, *TIME*(s_0)) predicate can be dropped, leaving only the original predicate without reference to time. This may also be done if the interval of a *HOLDSOVER* contains the interval under discussion. Second, this notation is often notationally simpler:

HOLDSDURING(*cs*,$[t_1,t_2]$)

is cleaner to read and compute on than

$\exists\, t\, (t_1 \leq t \wedge t \leq t_2 \wedge cs(t))$.

However, $cs(t_0)$ is slightly cleaner than *HOLDSAT*(*cs*, t_0).

We have been developing a computationally feasible basis for reasoning within the temporal logic outlined above. Appendix C demonstrates a program that manipulates imprecise temporal quantities over PROCESSES and EVENTS. We can infer selected aspects of a state at time t within a history H (i.e., *STATE*(t,H)) in terms of last-stated values and relevant changes occurring within a finite number of intervening circumstances.

*Alternatively, an interval argument could be represented by two time point arguments, giving *HOLDSOVER*(*cs*, t_1, t_2).

C. States and Circumstances

A **STATE** represents simultaneously true conditions and consists of a set of conditions that are true with respect to the contextual situation(s) the STATE is intended to model. A STATE s has an associated IMPLIED STATE $IS(s)$ that is the transitive closure of inferences possible from s (i.e., the complete set of all conditions either in s or derivable from s).

While most planning systems have been concerned with discrete state changes caused by operators, we feel it is necessary to model processes. We shall use the term **PROCESS** to describe situations in which conditions vary as a function of time. A falling ball would be a PROCESS in which the position (and speed) of the ball changes with time. At any instant, the position of the ball is a fact in the STATE at that instant. At any later instant, the STATE will have changed because the position of the ball has changed.

Every STATE occurs within the context of a **CIRCUMSTANCE**. A CIRCUMSTANCE c is accepted as true over an associated interval of time $[T_{initial}(c), T_{final}(c)]$. A CIRCUMSTANCE c consists of an initial state $INIT(c)$ which was true at time $T_{initial}(c)$, a set of (active) PROCESSES $PROC(c)$, and the set of planned activity with the state changes they cause, $EVENTS(c)$. Given a CIRCUMSTANCE c, the state may be determined for any time within the associated interval of time, presuming that the components of the CIRCUMSTANCE are all correct.

A CIRCUMSTANCE combines the representation of a set of simultaneously true conditions with a representation of the way those conditions are changing over a specified time interval. Our use of CIRCUMSTANCES and the assumption as to the nonempty property of time intervals will allow us to avoid problems that arise when reasoning is based solely on conditions true in instantaneous states, as discussed by McDermott [41]. By directly representing change as PROCESSES and EVENTS, we can reason about conditions that are true at any instant while bearing in mind the aspects of those instantaneous states that are currently changing. The maintenance of truth of a condition over some time interval can be expressed as a CIRCUMSTANCE wherein the condition is an element of the INITIAL STATE and is not affected by the PROCESSES or EVENTS of the CIRCUMSTANCE.

Control of the execution of planned activity will depend upon relationships among sequences of consecutive CIRCUMSTANCES. We say that CIRCUMSTANCE c_1 directly precedes CIRCUMSTANCE c_2 if $T_{final}(c_1) = T_{initial}(c_2)$. In this case, c_2 directly succeeds c_1 as well. We say that two CIRCUMSTANCES are consecutive if one of them directly precedes the other.

D. Histories

In order to reason effectively about present and past situations, we must be able to refer to what beliefs were held to be true by the system at various times. For the current time (i.e., now), which we call ***CT***, our knowledge not only includes statements about the present time ("The sun is shining") but also statements about past time ("The moon was full last night") and beliefs about the future ("The sun will rise tomorrow").

We shall define a HISTORY as a sequence of consecutive CIRCUMSTANCES c_0, c_1, . . ., c_n such that c_i directly precedes c_{i+1} for $0 \leq i < n$. A COMPLETE HISTORY over the interval $[T_{begin}, T_{end}]$ is a history such that $T_{initial}(c_0) = T_{begin}$ and $T_{final}(c_n) = T_{end}$. Given a COMPLETE HISTORY H over an interval and a time t within the interval, ***CSTANCE(H,t)*** is defined as the CIRCUMSTANCE c_i of H such that $T_{initial}(c_i) \leq t < T_{final}(c_i)$, being the CIRCUMSTANCE accepted as true at time t in history H. Similarly, ***CSTANCE(H, $[t_1,t_2]$)*** is the set of CIRCUMSTANCES c_i of H accepted as true at some time within the interval $[t_1,t_2]$. We define ***STATE(H,t)*** to be the state at time t as can be determined from *CSTANCE(H,t)*.

Since we do not know the true state of the world and only have beliefs about what is true, we shall describe the HISTORY OF CIRCUMSTANCES (that we believe are true) up to a particular time as an ***EXPECTED HISTORY****. For the current time *CT*, this history is called the ***CURRENT EXPECTED HISTORY***, or ***CEH***. The *CEH* represents (assumed) past, current, and (expected) future circumstances.

With each time in the past, $t \leq CT$, we associate an ACCEPTED EXPECTED HISTORY ***AEH***(*t*) that is equal to what the *CEH* was at that time. For future times, $t > CT$, we shall specify the *AEH*(*t*) to be simply *CEH*. *AEH*(*t*) is the complete history

*This use of "expected history" is closer to the use of "chronicle" by McDermott [41] than the use of "history" by Hayes [30]. The difference is that we are considering as part of our HISTORY only what we believe or know at the time, as opposed to what actually is true in the world. We have chosen the name EXPECTED HISTORY because most of the statements in a HISTORY would be about the past, but it also includes events that we expect to happen. Thus, it represents what is expected to be history at some future time.

of the WORKING PERIOD from the perspective of time t. Once a time t has become past, *AEH(t)* is fixed. Information received at a later time may indicate that *AEH(t)* is in error, either because the future did not happen as expected or beliefs about the past have changed. The new information must be incorporated into an updated *CEH*; however, *AEH*(t) does not change. Even if a system does not retain the *AEH*s, the concept is useful in describing why certain actions were taken in the past. If the *AEH*s are retained or are constructable, the information may be useful in determining what beliefs were wrong and why some expectation failed to materialize. This may be especially useful with a truth maintenance system. The set of *AEH*s is collected into a history of histories ***HH***. Each history is accepted over an interval of time that ends with the arrival of new information to the system about the state of the environment. We define the function ***HISTORY*** such that *AEH*(t)=*HISTORY*(*HH*,t).

IV ABSTRACT MODEL

This section contains an abstract model of the problem of interest. (See Appendix A for an example of this model.) The abstract model developed in the previous interim report reflected a view of plans as a sequence of events over time. Plans can be considered in two ways. If a finished plan is examined, especially if scheduling is of interest, it can be seen all at one level, flowing along with time (although there may be several parallel branches at once).

On the other hand, if a plan is examined from the perspective of developing the plan, it will probably develop hierarchically. The previous model was developed to allow monitoring of the plan as it unfolds. However, this is not a convenient view when the plan itself is generated or when it is modified during replanning. Proper planning is best achieved by top-down, recursive planning. Only in the simplest domains can a plan to achieve a goal be found without the necessity of finding plans for subgoals.

We have improved upon the earlier abstract model by providing the structure of a plan to allow recursive goals and subgoals without losing the ability to follow the plan linearly through time. When a plan has been completely formulated and all details filled in, the time events can be extracted, forming a linear list (still providing, of course, for activities occurring in parallel).

The following abstract model reflects these improvements.

A. Context

CONTEXT:
- A WORKING PERIOD
- An ENVIRONMENT
- STATE AXIOMS

All activity of interest is assumed to take place within a finite period of time, called the WORKING PERIOD. The WORKING PERIOD is represented by an interval of time specified as $[T_{begin}, T_{end}]$.

The ENVIRONMENT constitutes the conditional component of an execution context.

An ENVIRONMENT is defined by an interpretation over a set of relations, functions, and their domains. A relation is a named set of relation domain tuples. A relation domain is specified by a set of objects and associated object names (i.e., constant symbols). For example, a simple ENVIRONMENT for a class of warehouse control problems can be described by specifying the following relations: *AT*(*obj*, *loc*), *ON*(*obj*, *obj*), *ABOVE*(*obj*, *obj*), *NEXTTO*(*rob*, *loc*), *HAVE*(*rob*, *cart*), *IN*(*obj*, *cart*), where *obj* ranges over objects stored at the warehouse, *rob* over working robots, *cart* over available handcarts, and *loc* over possible warehouse locations. We assume the names associated with elements of these domains are of the form OBJ_i, ROB_i, $CART_i$, and LOC_i respectively, for positive integers i. A function maps its argument domains to a relation domain. A function domain is either a relation or a relation domain. A function *LOC*(*obj*) can be defined to equal loc_1 such that $AT(obj, loc_1)$[*] (i.e., $IN(obj, cart_1)$) exists).

Defining an ENVIRONMENT is equivalent to specifying a relational data base [27]. Each domain of a relation has an associated attribute name. With each relation we specify a subset of attributes to be key domains; for each instance of a set of key values there can exist at most one entry (tuple) in any instance of the corresponding relation. Thus, functions can be written in terms of relation and key arguments. Such functions would be akin to a combination of selection and projection operations in a relational algebra [10]. For example, *LOC* selects a tuple from the *AT* relation that is specified by a value for the key object attribute and returns the associated location value.

All preconditions and effects of aspects of planned activity are expressed in terms of condition statements over an ENVIRONMENT. A ground instance of a relation element (i.e., a tuple of constants from a given relation) is called a condition. A condition statement is either a condition or is formed from conditions by the introduction of domain quantifiers and variables and by the use of logical operators *NOT*, *AND*, and *OR*. We assume the existence of the pseudorelation *EQUALS*(*x*,*y*), which is true (i.e., includes the tuple (*x*,*y*)) if *x* and *y* are identical entities. For example, a condition statement indicating that OBJ_1 and OBJ_2 are at the same location can be expressed as the following:

$$AND(AT(OBJ_1, x), AT(OBJ_2, y), EQUALS(x, y)).$$

We let *AND* and *OR* take an arbitrary number of arguments for ease of expression.

In planning and controlling planned activity, we are concerned with the (believed)

[*] *NIL* if no such tuple exists

truth of condition statements at specific points in time or over given time intervals. A state s consists of a set of conditions that are considered to be simultaneously true at some point in time $TIME(s)$. As such, a state is an instance of the relational data base defined by the ENVIRONMENT that is consistent with previously specified STATE AXIOMS.

STATE AXIOMS express implications and constraints among condition statements that are true within a given state or within certain temporally related states. If we adopt the relational data base framework, uniqueness of certain nonkey values is implicit. For example, if the object attribute is made a key of the AT relation, an object can be at no more than one location in any state. Such constraints need not be expressed explicitly by STATE AXIOMS. STATE AXIOMS that express implications of, or constraints upon, conditions of a single state take the form

$HOLDSAT(cs, t) \Rightarrow HOLDSAT(cs', t)$,

where time t is understood to be universally quantified. Other STATE AXIOMS may express relationships among conditions at different points in time or holding over intervals of time. This allows the representation of changes in the ENVIRONMENT that are triggered by coexisting conditions and are not directly caused (intended) by the activity of agents.

Preconditions and effects are also expressed in the temporal logic. For example, the preconditions of an action scheduled to begin at time t may include the following requirement:

$HOLDSAT(EXISTS(x, OR(AT(x, LOC_1), AT(x, LOC_2))), t)$,

stating that there must exist some object at location LOC_1 or LOC_2 at time t.

One aspect of the ENVIRONMENT deserves special attention, that of RESOURCES. RESOURCES are material entities used in the performance of planned activity. RESOURCES may be reserved for, allocated to, and possibly consumed by a particular activity. Resource management is a crucial component in the effective control of planned activity. As such, we represent material resources as separate aspects of the conditional context. RESOURCES are represented in terms of use-related properties. In addition, each resource has an associated CURRENT USE HISTORY that, at any time within a WORKING PERIOD, indicates its past, current, and anticipated modes of use during the WORKING PERIOD. The CURRENT USE HISTORY associated with each resource (type) provides the indexing needed for updating allocations effectively.

The activity planned for a given WORKING PERIOD will be represented from two perspectives. One is as a hierarchy of GOALS, PLANS, and TASKS. The other is as a history of required conditions and a history of conditions expected to occur according to an extant hierarchy. As such, planned activity is represented in terms of the following entities.

B. Activity

ACTIVITY:
- A Set of GOALS
- A Set of PLANS
- A CONDITIONS HISTORY
- An EXPECTED HISTORY

Planned activity is naturally described in terms of GOALS and PLANS. Relational structures among these entities represent reasons, conditional dependencies, and other restrictions (e.g., relative constraints on execution time). A GOAL represents a set of conditions that the planning system desires to be established in the execution context over some specified period of time.

GOALS and PLANS will always exist as matched pairs. A PLAN for a GOAL is only relevant within the full CONTEXT. The PLAN associated with a GOAL may not be sufficient to achieve the GOAL without the actions that are specified in other PLANS within the CONTEXT. In particular, a GOAL may have an empty PLAN if other PLANS will achieve the GOAL. The CONTEXT embodies the associations between a GOAL and the PLAN to achieve it.*

Corresponding to a symbolic execution (i.e., simulation) of the scheduled TASKS of currently planned ACTIVITY is a CURRENT EXPECTED HISTORY *CEH* that reflects changes expected to occur over the WORKING PERIOD. Events in this history reflect the initiations and completions of scheduled TASKS as well as the implications of any applicable STATE AXIOMS. Simulation of a set of scheduled TASKS from a given beginning state produces a sequence of time-consecutive circumstances constituting *CEH*. Through application of the axioms of the temporal logic, expected values of condition attributes can be deduced from this history for any point in time within the WORKING

*We presume the existence of some interconnections such that the path from any item, say a STEP, to any other item, say the PLAN in whose METHOD/SCHEDULE the STEP is contained, can be traced.

PERIOD. This capability serves as the basis for determining whether new information is consistent with current expectations.

C. Goal

GOAL:
    GOAL CONDITIONS
    A RATIONALE

GOAL CONDITIONS are expressed as statements in the temporal logic; as such they indicate condition statements to be true at associated times or intervals of time. A GOAL's RATIONALE indicates the role of the GOAL in realizing either the top-level goal (what the planner was ordered to achieve), or that it is an essential subgoal in achieving some other goal, or that it is necessary to satisfy a general, recurring objective (e.g., maintaining adequate supplies). Satisfaction of several GOALS may be necessary to attain a global objective.

D. Plan

PLAN:
    PLAN CONDITIONS
    PLAN EFFECTS
    A METHOD/SCHEDULE

The PLAN CONDITIONS contain what have been called "preconditions" [22] for the plan. That terminology is too limited for this application as there may be conditions that must hold at some point in time after the plan execution has commenced, but not necessarily before then. Likewise "postconditions" may become valid before the PLAN terminates, so we use the term "EFFECTS." The PLAN CONDITIONS are the (often, but not necessarily) minimum set of conditions required for its METHOD/SCHEDULE to succeed. The PLAN EFFECTS describe the plan's effects upon the ENVIRONMENT. This includes all the effects of the subplans contained within this plan.

The METHOD/SCHEDULE of a PLAN consists of a procedural network [32] of STEPS or GOAL/PLAN pairs. Any GOALS occurring within a METHOD/SCHEDULE are subgoals of the main GOAL associated with the PLAN. These would then recursively have their own PLANS with METHODS/SCHEDULES. No PLAN is complete until each node in the procedural network of the METHOD/SCHEDULE of that PLAN is either a STEP or a GOAL whose PLAN is complete. The SCHEDULE part of a METHOD/SCHEDULE can be extracted from the STEPS below it.

A PLAN, as defined here, represents a marriage of (1) the procedural-network planning technology developed in artificial intelligence with (2) the PERT/CPM technology developed in project management. The resultant representation includes both functional and temporal relationships among the component TASKS. Steven Vere's planning system under development for NASA [57] adopts a similar approach in scheduling the behavior of planetary space probes.

The METHOD of a PLAN represents relationships among the preconditions and EFFECTS of TASKS associated with STEPS or SUBGOALS of its METHOD. The SCHEDULE of a PLAN consists of the time interval $[T_{start}, T_{over}]$. T_{start} is equal to the earliest (least) starting time of the STEPS and SUBGOALS in the METHOD. T_{over} is equal to the latest (greatest) expected completion time of the STEPS. This time interval represents the extent of time within the WORKING PERIOD during which STEPS of the PLAN are expected to be executed. A PLAN's SCHEDULE also has slack measures determined with respect to the ESTABLISHED PERIOD of its associated GOAL.

When a PLAN is completely filled out, its hierarchical structure could be removed by replacing each subgoal in the procedural network of the top-level METHOD/SCHEDULE with the procedural network of the METHOD/SCHEDULE of the subgoal. This is repeated until all subgoals are removed and the METHOD/SCHEDULE of the top-level PLAN is a procedural net of STEPS.

Flattening of the PLAN hierarchy results in the loss of information and therefore should not actually be carried out. It may be useful to reachieve only some subgoals during replanning after an unexpected event rather than having to determine what must be done to guarantee accomplishment of the overall goal. If the PLAN has lost the structure of the subgoals, the whole plan may have to be redeveloped.

E. Step

STEPS constitute nodes of the procedural network, while arcs represent precondition dependencies among TASKS associated with the STEPS. Each STEP has an associated TASK. A TASK may be associated with one or more STEPS from one or more PLANS. A STEP is represented in terms of the following aspects.

STEP:
  A TASK
  A SCHEDULE

The SCHEDULE of a STEP reflects the starting time and expected duration of its associated TASK. In addition, various measures of slack (or float) [57] are indicated. Each measure of slack indicates a bound on rescheduling a STEP without any corresponding need to reschedule other aspects of the PLAN. One measure may indicate how much a STEP may be pushed back (or ahead) without causing other STEPS to be rescheduled. Another may indicate similar bounds if the overall beginning or ending time of a PLAN is not to be affected (even though other STEPS may have to rescheduled within the PLAN's METHOD/SCHEDULE).

A STEP exists within the procedural network of the METHOD/SCHEDULE of a PLAN. That network contains information about the predecessors of this step (usually validating the latter's preconditions) and the successors of this step (awaiting the preconditions supplied by this step).

F. Task

A TASK represents a segment of activity that is normally executed without interruption. Each TASK is represented by several aspects:

TASK:
  A SCHEDULE
  A METHOD
  INSTRUMENTS
  PRECONDITIONS
  EFFECTS
  PLAN STEPS

The SCHEDULE aspect represents information as to scheduled starting time and duration of a given TASK. Since the duration of a task may depend on other factors, estimates are included for the minimum, maximum, and expected durations for successful task execution.

The METHOD aspect serves to describe execution of the TASK. A TASK may be primitive (an ACTION) or compound. An ACTION is a TASK having an empty (undescribed) or functional METHOD. A functional METHOD describes the dynamics of the state change culminating in an ACTION's effects as a function of its SCHEDULE. The METHOD aspect of a compound TASK is a 'program' specifying an execution sequence of subordinate TASKS. Duration aspects of the SCHEDULE associated with a compound TASK can be determined from those of its subordinate TASKS in a straightforward manner when independence of subordinate task durations is assumed [57].

The METHOD of a PLAN, through relations among its STEPS, places constraints upon the SCHEDULES of its STEPS. STEPS of a PLAN typically need not be performed consecutively. Some may be accomplished concurrently; some, by establishing the preconditions of other steps, must be completed before the latter can begin. Their flexibility, interruptability, and possible concurrency are what primarily distinguish the METHODS of PLANS from those of TASKS.

An INSTRUMENT provides for a necessary capability in the execution of a TASK. An INSTRUMENT role is filled by one or more assigned (allocated) RESOURCES. Each INSTRUMENT is specified in terms of constraints on use-related properties associated with resources. These constraints define resources that can be used to fill an instrument's role in TASK execution . Resources currently assigned to a given INSTRUMENT are noted as well.

A PRECONDITION is a condition that is not provided by the TASK itself, but is necessary for the successful execution of a TASK. A PRECONDITION is expressed as a statement in the temporal logic described above. An ACTION has a prespecified set of PRECONDITIONS. A compound TASK's PRECONDITIONS are defined in terms of PRECONDITIONS of its subordinate TASKS, being equal to the union of the PRECONDITIONS of its subordinate TASKS minus those provided within the METHOD itself. These can be determined by a process known as regression [44]. PRECONDITIONS generated by subordinate TASKS need not be established until execution of those subordinate TASKS commences.

The EFFECTS of a TASK represent the state change resulting from successful execution of the TASK. The EFFECTS indicate both a set of conditions and the time at which such conditions are established during TASK execution. These are conditions that are not subsequently consumed by the TASK; they are determined by a process of progression, similar to regression for PRECONDITIONS. EFFECTS are expressed as statements in the temporal logic.

G. Activity Reports

We assume that the PLANS are sufficient to satisfy the GOALS, within the constraints of WORKING PERIOD and RESOURCES, starting from the beginning state BS (i.e., the state at time T_{begin}). Unfortunately, we also must assume that the execution context cannot be well predicted. However, indications as to prevailing context situations can be obtained. As the WORKING PERIOD progresses, information is gathered in the form of ACTIVITY REPORTS of the following types.

ACTIVITY REPORTS:
- ENVIRONMENT REPORTS
- RESOURCE REPORTS
- TASK REPORTS

Each ACTIVITY REPORT has a content and a time aspect. The content of a report is accepted as true as of its associated time aspect. ENVIRONMENT REPORTS contain information that is gathered by sensory processes. Such information may be obtained as a necessary side effect of TASKS carried out for other purposes. Other (e.g., reconnaissance) TASKS are scheduled for the exact purpose of acquiring certain types of contextual information. The content aspect of an ENVIRONMENT REPORT takes the form of a set of contextual conditions. RESOURCE REPORTS indicate changes in the usage modes of RESOURCES. Most RESOURCE REPORTS are generated in conjunction with TASK REPORTS. TASK REPORTS indicate starts, completions, and outcome assessments (e.g., success, failure) of TASK executions.

The first stage in determining the significance of a newly obtained report having content C and time t is to evaluate the consistency of C with respect to conditions in $STATE(t, CEH)$ and happenings in $CSTANCE(t, CEH)$. CEH is the current expected history of the WORKING PERIOD; $CSTANCE$ retrieves the circumstance within which $STATE(t, CEH)$ occurs. This evaluation may indicate the need to bring that circumstance into conformance with the new report content C. If accommodation of history is required, the next stage is to determine those aspects of planned activity that are affected by the necessary changes. This stage involves processes that reason according to precondition dependencies and time slacks of PLAN METHODS and others that simulate selected, scheduled TASKS so as to propagate the consequences of the new information to all places concerned. Finally, the full import of new information is realized through interactive processes that serve to adapt existing PLANS and GOALS to opportunities and difficulties engendered by the new report.

ACTIVITY REPORTS that necessitate history accommodation result in a history of histories *HH* (see Section III-D). Each history is accepted over an interval of time that ends with the arrival of one or more such reports (which is often after the times of those reports). Each element of *HH* has associated with it the ACTIVITY REPORTS and ACTIVITY CHANGES that gave rise to its generation. We expect that access to *HH* will provide a basis for analyzing and discussing the course of a WORKING PERIOD after it has unfolded.

The objective of execution time control and replanning is to respond to new reports in a manner that maximizes the value of GOALS satisfied during the WORKING PERIOD. We assume that initial PLANS attempt to optimize use of the WORKING PERIOD. Optimization procedures require GOAL evaluation guidelines (e.g., global strategy). A GOAL's RATIONALE is used to determine its importance with respect to such guidelines.

H. Replanning Reports

As information is gathered and its significance ascertained, activity changes may be proposed in the form of REPLANNING REPORTS of several types:

REPLANNING REPORTS:
- Rescheduling an Existing TASK
- Canceling an Existing TASK
- Introducing a New TASK
- Introducing a New GOAL and Associated TASKS
- Canceling an Existing GOAL and Associated TASKS

The adoption of appropriate activity changes ultimately depends upon the ability to comprehend the significance of ACTIVITY REPORTS and subsequent REPLANNING REPORTS with respect to current PLANS and scheduled TASKS. It is this ability, together with those aspects of a system that make this ability possible, that constitutes the focus of our design and development research.

V SYSTEM REQUIREMENTS AND DESIGN

A. System Layout

In this chapter we discuss the requirements for some portions of a system that can provide support for monitoring previously planned activity and adapting those plans as necessary. To put this in perspective, we first give a brief sketch of how such a system might be put together and then discuss the subsystems and their logic in more detail. These subsystems are:

- a data base system,
- a monitor (consisting of a deductive component and a simulation component), and
- a human interface (not covered in this report).

The current-plans data base contains all the data about the plans, including expected events and states, as well as information about the state and history of the world. Obviously, the data base does not reflect the actual situation in the world but only reflects the beliefs of the system based upon whatever data it has received and digested. This component is critical because the execution of the system depends entirely upon this information.

Controlling the execution of the system would be a monitor. This segment would be responsible for accepting information external to the system and acting upon it. After receiving status reports or other information about the state or history of the world, this segment would be responsible for ascertaining the import of these data. It would cause the data to be compared with present knowledge about the world (including what is expected by the plans), would merge this knowledge into the data base, and would note any problems these data present for the plans. The monitor would then report these problems to the operator. In some cases, it may be able to suggest appropriate changes to the plans but, since we are not attempting to do planning within this system, the operator would normally suggest alternative plans. The monitor would take these and compare them to the data base to determine if they are acceptable.

Because of the complexities of the replanning domain, we envision that any replanning system will be run interactively to allow human operators to make the final decisions. So another part to the system, which is not covered in this report, would be the human interface to the system, sitting between the system monitor and the human operator. It would be responsible for representing the knowledge and conclusions of the system to the operator and also for accepting his queries and commands to the system.

The monitor would call upon two major subsystems: the deductive component and the simulation component. These would do much of the work of the system. The simulation component could take the present or some anticipated future state of the world (as represented in the data base) and would extrapolate it forward through time. By using this extrapolation the monitor can determine whether a proposed action is wise or whether a possible situation presents a problem.

The deductive component is an essential portion of the system and is where much of the work will be done. Every piece of information that the monitor gets will need to be compared to the existing data base to see if it is consistent with expectations. This will be done by passing it to the deductive component and looking for contradictions. When the monitor needs to know the truth of some formula it will generally call on the deductive component. When the deductive component returns, it will not only report whether it was able to determine the validity of the formula but also what beliefs and rules it used to determine the truth (or falsity) of the formula. From that would come a measure of the certainty of the formula based upon the certainty of the beliefs and rules used to determine it.

B. The Monitor

The most critical component of the entire system is be the monitor. The best laid plans and accurate status reports are useless unless they can be compared to determine what is the status of the plans. Have they been carried out properly? Is there some impediment to their future execution?

When information is received, it needs to be compared against the data base. This would be done by calling the deduction component to determine if the information causes any problems if added. If not, the monitor would add the information to the current-plans data base.

Problems come in two classes. One type is where the information apparently contradicts other information in the data base. If that other information was used in verifying that the plans will succeed, then we have the second kind of problem, possible plan failure. Some information may indicate that a plan has already failed to achieve its goals.

When a problem is detected, all its ramifications need to be discovered. The system needs to determine what other plans the failure of a particular plan step may have already affected or may affect in the future. If the cause of the failure can be determined, other plans that would be similarly affected should be discovered. If a condition that will be necessary for future action will not be satisfied, that future action must be assumed to fail and all its subsequent effects will not be achieved. By anticipating future problems when the initial problem is discovered, the system can aid in finding new plans to avoid as many of the difficulties as possible.

When new or alternative plans and actions are suggested, the monitor can use the deductive and simulation components to compare these to its world knowledge to determine if they are feasible. It can confirm that resources are available when needed; it can verify that scheduling is consistent; it can check that all conditions for the actions can be expected to be satisfied; and finally, it can verify that the goals will be achieved if the actions are carried out.

The person responsible for making decisions based upon this system will need to gather as much information from the system as possible. To this end, the system will need to be able to justify all of its conclusions if asked. In order to do this, it will be necessary for the system to keep track of how each assertion in its database was originated. Was it in the original database? If not, what other assertions were used to derive it and what was the justification for that derivation? How are those other assertions justified? What certainty does the system have for each of the beliefs? The human engineering of properly presenting this information as required is not within the scope of this project but the derivation information must be present within the system.

This system will need to reflect the real world as accurately as possible. However accurate it may be, there must be some margin for error. Errors may arise from erroneous information derived from intelligence reports or sensors—information that may be in direct contradiction to what the real situation is. Furthermore, the system will need to make assumptions about what is happening in the real world even when it has no direct confirmation that expected events actually occurred. It will adopt the belief that the expected event happened and will include the consequences in its data base.

If information is later found to be erroneous, the system must be able to recover without starting over. There may have been many deductions, resulting in assertions being added to the data base, that have taken place since it accepted the false data, some of which are valid and some invalid. The system must be able to recognize that it no longer accepts the false data as valid and recant all deductions based upon it.

The system may not be told that some information is invalid; it may learn it the hard way. If the system finds some contradiction by deducing some fact for which it already "knows" the negation of that fact, then it must be able to recover. If it can, it should determine which one of the two facts is wrong and should also determine what caused it to derive an incorrect assertion.

C. Simulation Component

While we feel that the deduction system is better suited for most things the replanning system will need to do, it may be useful to include a simulation component. A simulation system will calculate future situations from the present situation while deduction can also compute present constraints from future requirements as well as computing logical deductions that do not involve temporal aspects. Obviously deduction could be used for the calculation of future situations from the present as well, but it may only be efficient in projecting some predicates. Simulation may do a better job for projecting more details.

A simulation system would take the state of the world as described by the data base (or possibly the expected state of the world at some future time) and copy it into a simulation data base. Then it would simulate the events that might happen beginning at that point in time and correspondingly update its data base. As it runs, it might detect unforeseen problems that will need attention.

There are several problems with simulation. First, since the system can not know ahead of time what features may be important for the simulation, it will need to simulate what happens to all the features it knows about. It would need to simulate the weather, the positions of aircraft, defenses, troops, etc. But in order to be fast, it would be desired to minimize the features simulated.

Second, simulation will only consider one future track at a time. The results of the simulation only indicate one possible future and so prove the possibility of the result without indicating the certainty or even a probability of the results reflecting what really would happen. If the simulation is run many times, then some confidence could be gained about the likelihood of a particular situation arising.

However, this may be useful when used by the monitor in conjunction with the deduction system. When the deduction system tries to look too far into the future, it might require a longer time than a simulation system would (depending on the tightness of the search). In such a case, the simulation system could be called upon to determine if a particular event might occur. The simulation system could say that it would (but could not say that it definitely would not).

Of the simulation systems we looked at, the object-oriented simulation system from Rand [33] is coarse-grained enough to minimize the execution time as we would require.

The simulation structures of FOL [64] do not constitute a simulation system but do provide a way of tying simulation-like processes into deduction. It offers a method by which computation (like simulation) may be called upon to decide that a certain fact is true. The decision may not be found, in which case the system says that it does not know the truth of the fact. But if it can determine the truth, the deduction system can then use the fact in its proof. This turns out to be somewhat like the Theory Resolution we describe later.

We also considered what we call Worst-Case simulation. By this we mean simulating the possible future events in the world by having them follow the path that is most damaging to the goals we choose. For instance, if enemy troops are within two days travel of a particular position, then we will assume that they will reach that spot in two days, regardless of other conditions that may delay their progress or divert them entirely. This is not meant for use as a true simulation but rather to create bounds upon the possible range of events that may happen in the future. Thus, if we need to know if the enemy can reach a certain position in one day and we can deduce that they are at least two days away, we can avoid considering paths that include the enemy reaching the site. On the other hand, if we can not rule out their reaching the site, then simulation may be necessary to determine whether they will actually reach it.

D. Requirements for the Deductive Component

The performance of the deductive component, its power and speed, is critical. A real time, real world replanning system is very demanding. We present here the outline of a deduction system that we feel would be the best match for these demands. We shall be employing ideas from many different systems, bringing them together into one system. By combining the different parts, the deductive component will have enough capability to be used in the replanning system.

Unfortunately, having the capabilities to do the deduction is not enough. The deductive component will need to be fast enough to keep up with the real-time demands of the problem domain. These real-time constraints will be kept in mind as we present the system.

E. Capabilities of the Deductive Component

The deductive system will need to have at least the following capabilities:

- Applicable to expressions in a temporal logic.
- Accountable for its deductions.
- Able to recover from changes in its beliefs.
- Able to recover if it finds apparent contradictions in its data base.
- Effective on a large data base of information.

The deductive system will be applied to assertions in the current-plans data base. It follows then that it must be effective in dealing with expressions in the temporal logic presented in Chapter III. This will require that the deductive component be effective in processing the time intervals. Similarly it will need to be able perform spatial computations.

We have already indicated that the monitor will require keeping track of why information is added to the data base. Since it will need to know how a belief was derived, the deduction component will need to report that back to the monitor. Each deduction it makes must be traced.

As a belief is revised, the monitor will modify the data base. It probably will not physically delete entries that are no longer believed but will mark them in some way. The deductive system will need to ignore some of the data according to the belief in it.

The data base may very well have contradictions in it that have not yet been discovered. If the theorem prover detects one of these, it will need to report it back to the monitor. The monitor needs to know how the contradiction was found, just as it needs to know how proofs are done, so it can determine what assumptions may be invalid. After the monitor changes the beliefs in the data base, it may recall the deduction component again to verify that no other inconsistency was introduced.

Because the replanning system will need to know the state of the world to determine whether the plans are being, and will continue to be, executed properly, the data base of information will be immense. Because of this large body of data, the deduction system will need to be made as efficient as possible. It can not afford to get swamped by analyzing data that is not pertinent to the question it is addressing.

The problem with large numbers of assertions for a deduction system is the combinatoric explosion resulting from the relation of one assertion to many other assertions. In generating a deduction, each step has many different possible successive steps. The result is a search space that grows exponentially as the proof gets deeper. The exponent is a function of the number of assertions available. This suggests that simulation could be used profitably when a long deduction involving many assertions is required. On the other hand, simulations are just approximations of the future situations and may not reflect all possible outcomes. By using deduction and simulation where they are most effective, the whole system can be stronger.

F. Type of Deduction System

There are several types of deductive engines that could support these capabilities. Among the possibilities are production systems, natural deduction theorem provers, and the various flavors of resolution-based systems.

Production systems, such as those used in Wesson [62] and Cohen and Grinberg [14] and expert systems (e.g., Georgeff and Bonollo [28]), are often associated with problem-solving systems. A production system works by checking its list of rules (productions) until it finds one whose conditions for applicability are true at which point it applies the rule. Control information can be kept either in the program selecting the rules or in the rules themselves. The world knowledge is encoded only in the rules and not in the program driving them. The rules can encode detailed world knowledge. But, for general deduction, a production system might have to find, say, a rule of *modus ponens* and apply it.

Natural deduction theorem provers, such as Bledsoe and Tyson [4], are better suited for general deductions. This type of deductive engine attempts to find proofs in a fashion similar to how a person might try it. For instance, formulas are kept in their original form rather than being split into clauses, goals are split into subgoals, and the proof proceeds in a positive fashion (neither assertions nor conclusions are negated). It is easy to build on top of natural deduction systems, so constructing, say, a truth maintenance system (TMS) using natural deduction should present no problem. However, they also tend to have a relatively high amount of overhead and are not as well suited to making thousands of deductions as are resolution-based systems.

In the past twenty years, many types of resolution-based systems have been developed. A style of programming, logic programming, has developed on top of one subset of resolution, Horn clauses. PROLOG [13] is a logic programming language whose execution proceeds by making logical deductions. This language has been chosen by the Japanese as one of the basic elements of their fifth generation computer project. The speed of Prolog in doing these deductions is quite fast—David Warren's implementation of PROLOG on DEC machines has a speed of 20,000 to 40,000 logical instructions per second. Although PROLOG is quite fast, the deductive engine for our system would have to be built on top of it to encode the features we would need such as a TMS. So, while the base language might perform resolutions quickly, the effective rate would be much less. However, if the Japanese do eventually build a machine that can execute at 1,000,000 logical instructions per second, the complete system would be extremely fast.

Connection-graph resolution appears to be a good choice for dealing with a very large data base of information. In resolution, each step consists of taking one expression from the data base and combining it with another that has a complementary literal. That is, an expression that contains the literal A would be resolved against another that has the negation of A in it. Once the first expression is chosen, an expression to resolve it with must be found. Rather than searching through the entire data base, connection-graph resolution maintains a set of pointers from any instance of a literal to the instances of complementary literals it could resolve against.

Besides reducing the time searching for complementary literals, connection-graph resolution also restricts resolution, reducing the number of resolution operations that are permitted. Furthermore, in looking for a proof, the system can perform graph-searching by following along the links between complementary literals. Only after the system is satisfied that a proof may exist along that path would it actually perform the resolutions and create the newly derived formulas.

One of the complaints often made about resolution is that it requires its input formulas to be converted into clauses rather than retaining the original logical connective. Not only is the clause form unnatural for someone trying to work with it, but this conversion may lead to loss of heuristic clues on how a formula may be best used. While $P \Rightarrow Q$ and its clause form, $\neg P \vee Q$, are logically equivalent, the former suggests chaining while the latter suggests case analysis. Furthermore, conversion to clause form typically increases the redundancy in the system. The formula $A \Rightarrow (B \wedge C)$ would become two clauses $\neg A \vee B$ and $\neg A \vee C$.

It is no longer necessary to reduce a formula to clause form as nonclausal versions of resolution have been developed. This improvement is not without its drawbacks though. Because the structure and logical meaning of each term of a nonclausal formula is more complex than for clauses, the operations on the formulas are more complex. However these problems may be worked out. For more details on a nonclausal connection-graph resolution system, please see Stickel [53].

G. Theory Resolution

Resolution has been refined many ways since it was originally introduced. Theory resolution, developed by Mark Stickel [54], is a somewhat different refinement. It provides for a procedure, possibly different from resolution, to help determine the inconsistency of a set of clauses (it is extensible to nonclausal formulas) according to some theory. In total theory resolution, this other procedure (a decision procedure) would be responsible for operating on the set of predicates in that theory. For instance, for a theory of partial ordering, the procedure would be responsible for determining the conditions for inconsistency of a set of clauses containing the inequality predicate "<." Once this decision procedure has determined conditions (substitutions for free variables) necessary for the inconsistency of the set of clauses containing the predicates it is responsible for, a set of clauses containing none of these predicates is generated from the original set. This set can then be given to the resolution theorem prover.

Theorem.[*] (Rule of inference for total theory resolution.) Let S be a set of ground clauses and P be a set of predicates (i.e., the predicates in the theory). Let S_P be the set

[*]The theorems and definitions are from Stickel [54].

of all clauses of S containing only predicate symbols in P. Let $S_{\bar{P}}$ be the set of all clauses of S containing only predicate symbols not in P. Let W be $S - S_P - S_{\bar{P}}$. Let W_P be the list of clauses C_i formed by restricting each clause in W to just the predicates not in P. $W = \{C_i \vee D_i \mid 1 \leq i \leq n\}$. Let X be the set of all clauses of the form $D_{i_1} \vee \ldots \vee D_{i_m}$ where $C_{i_1}, \ldots, C_{i_m}$ are all the clauses of W_P in a minimally inconsistent set of clauses from S_P and W_P. Then S is inconsistent if and only if $S_{\bar{P}} \cup X$ is inconsistent.

Definition. Let $C_1, \ldots, C_m$ be nonempty clauses and $D_1, \ldots, D_m$ be clauses such that each $C_i \vee D_i$ is in S and every predicate in C_i is in theory T and no predicate in D_i is in theory T. Let $\sigma_{11}, \ldots, \sigma_{1n_1}, \ldots, \sigma_{m1}, \ldots, \sigma_{mn_m}$ be substitutions such that $\{C_1\sigma_{11}, \ldots, C_1\sigma_{1n_1}, \ldots, C_m\sigma_{m1}, \ldots, C_m\sigma_{mn_m}\}$ is minimally T-inconsistent. Then $D_1\sigma_{11} \vee \ldots \vee D_1\sigma_{1n_1} \vee \ldots \vee D_m\sigma_{m1} \vee \ldots \vee D_m\sigma_{mn_m}$ is a ***total theory resolvent*** from S, using theory T.

Presburger arithmetic (integer addition and inequality) is a theory that is a candidate for total theory resolution. In our application domain, if we restrict our time points to be nondense (e.g., limit them to be in terms of an integer number of seconds or even nanoseconds), then we can use total theory resolution to separate out all the predicates of inequality and addition of time points. The system would pass to the separate decision procedure all the clauses concerning temporal relations. It would then combine the resulting clauses with those remaining and perform its deductions on the reduced problem. By effectively dividing the original set of clauses into two groups and solving one, the explosive potential of the search for a solution due to the branching factor (based upon the number of potential matches) is significantly reduced.

There is another restricted form of theory resolution, partial theory resolution, that is less demanding on the decision procedure for the theory. Rather than working on sets of clauses, the decision procedure only needs to determine a complete set of substitutions and conditions for the inconsistency (according to the theory) of any pair of literals. From this we can generate *T-resolvents* of clauses containing predicates of the theory T. As a simple example, $\neg Fighter(P)$ is a T-resolvent of $B52(P)$ and $\neg Bomber(x) \vee \neg Fighter(x)$ for a taxonomic theory containing $B52(x) \Rightarrow Bomber(x)$. There is no need for the system to derive or retain the clause $\neg B52(x) \vee Fighter(x)$. In effect, each deductive step is more powerful, allowing solutions to be found in fewer steps and resulting in a smaller search space.

Definition. Let A and B be two literals. Then $\langle E,\sigma\rangle$ where E is a clause nd σ is a substitution is a ***T-match*** of A and B if and only if $T\vdash\neg A\sigma \vee \neg B\sigma \vee I$ but not $T\vdash\neg A\sigma \vee E$ nor $T\vdash\neg B\sigma \vee E$.

Definition. Let A be a literal and $A \vee C$ be a clause and let σ be a substitution such that $T\vdash\neg A\sigma$. Then $C\sigma$ is a ***T-resolvent*** of $A \vee C$.

Definition. Let A and B be the nonempty clauses $A_1 \vee \ldots \vee A_m$ and $B_1 \vee \ldots \vee B_n$, let $A \vee C$ and $B \vee D$ be clauses, and let $\langle E_{ij},\sigma_{ij}\rangle$ be T-matches of A_i and B_j. Then $C\sigma \vee D\sigma \vee E\sigma$ is a ***T-resolvent*** of $A \vee C$ and $B \vee D$ where σ is the most general combined substitution of $\sigma_{11},\ldots,\sigma_{mn}$ and E is $E_{11} \vee \ldots \vee E_{mn}$.

For more details on theory resolution, please see Stickel [54].

H. Nonmonotonic Reasoning

With perfect knowledge, perfect plans could be made. With perfect plans, no plans would ever need to be modified. But our knowledge of the real world is incomplete at best and occasionally incorrect. Plans fail and we try to adapt.

A plan is a projection of future activity based upon present knowledge. If we learn something new about the world, our knowledge changes and we may realize the plan will fail. If our new knowledge is that some explicit fact in our old beliefs was wrong, then it is not too surprising that a plan would fail. Perhaps we believed that a plane had full armament only to find out later it did not. A subtler problem arises when we gain new knowledge that does not contradict any previous facts but does lead us to different conclusions. An example might be the discovery of enemy planes along a flight path that had been thought to be clear.

There has recently been considerable interest in nonmonotonic logic [40, 42, 39, 64]. Classical logic is monotonic: facts deduced before the addition of more data are still deducible afterward. Whatever is deduced need never be retracted. In a nonmonotonic logic, deductions may not be valid after more information is added. Extending a theory may invalidate previous theorems.

One obvious use of this is for belief revision. If we believe some fact, then it is natural to use it as a basis for deriving other facts. If we believe today to be Monday, we go to work since we always do that on Monday. If we later determine that our belief was wrong, that it is actually Sunday, our conclusions and actions must change. (Notice that the truth of "Today is Monday" does not really change in the real world. We use it as true in our deductions until we realize it was false.)

Another obvious use of nonmonotonic logic is default reasoning. If Polly is a bird, then we presume Polly can fly since this is the normal case for birds. If we later find out that Polly has its wings clipped, we must retract that deduction. We could add a qualification that only birds whose wings are not clipped can fly. But then Polly might be an apteryx, might have a broken wing, might be newly hatched, et cetera. This is the qualification problem [39]. We could continue to add qualifications but we would probably never have all of them. Furthermore, we would have to check all the qualifications each time we wanted to use the rule that birds fly.

All of the existing systems that reason about the real world use rules that could have many qualifications attached. These systems probably have some of the relevant qualifications for its rules but none will consider all the possible exceptions. Since our system will be operating over the real world, we will probably also be subject to the same problems. Likewise, we will encounter situations in which our belief of what is true in the real world changes. We will need to use some form of nonmonotonic logic.

I. Truth Maintenance

The Truth Maintenance System (TMS) of Doyle [16] is an implementation of a nonmonotonic logic system. In his system he keeps track of all the justifications for each belief the system has and these are propagated during deductions. Suppose statement S_1 is justified by the evidence E_1 and statement S_2 is justified by E_2. Then if the TMS derives statement S_3 from S_1 and S_2, the statement S_3 is justified by the conjunction of E_1 and E_2 along with the rule that derived S_3. As long as E_1 and E_2 are *in*, i.e. believed, then S_3 has well-founded (noncircular) support and so is believed and used. If, say, the belief in E_1 changes, the system no longer believes in S_1 nor S_3. Thus the proper belief of the statements is maintained even in the face of changing hypotheses.

The methodology of the TMS seems to be right for our system since we are faced with two major problems:

- We must assume particular states for many features of the real world for which we have no direct, current knowledge (default reasoning).

- Invariably, some of our previous knowledge or assumptions will be proven wrong and we must recover without restarting.

The first of these problems is resolved by the default reasoning capability of the TMS. One of the justifications that can be used for a statement is that some other statement is (or is not) believed. For instance, we may have the statement that a bird can fly. One justification for that might be that there is no proof that the bird can not fly. Thus if we know Polly is a bird and we have not found that Polly can not fly, we are justified in believing that it can fly. If we later find that Polly's wings are clipped and we have a rule that birds with clipped wings can not fly, we will add the statement that Polly can not fly. But adding that statement removes the justification from the statement that Polly can fly. The database is consistent; the apparent inconsistency disappeared.

The second of these problems is exactly what the TMS does: it automatically tracks what beliefs are based on what other beliefs and can properly update the system as beliefs change.

A TMS can do even more than this. If the system finds that a belief is in contradiction with other beliefs (which are perhaps reports of the state of the real world), then it can collect the set of beliefs upon which the erring belief is based. If exactly one of these is a default-type belief (i.e., one based upon the nonexistence of another belief), then that is probably incorrect. The system can then retract that default belief and, in doing so, retract any other beliefs based upon the apparently faulty belief. The spread of the error can be limited in this way.

J. Uncertainty

Almost all of the knowledge we have about the world is uncertain to some extent. We tend to rank observations as being fairly certain, depending on how good our senses are. But even our senses may fail us. When information comes from outside, it is subject to the same possibility of originally being wrong and is also subject to a failure of communication. This would include both data being lost or partially damaged as well as a misunderstanding about the data's content.

People often lessen the degree of uncertainty by gathering supportive evidence from several sources. As the degree of support increases, the certainty of a belief likewise rises. At some point, we stop looking for further evidence for the belief but will note any contrary evidence. A belief is strongly believed when almost any contradicting evidence would be discarded as probably in error rather than causing a reexamination of the belief.

While people are rather adept at maintaining their beliefs and adjusting them appropriately according to the evidence (although not everyone is that wise all the time!), computer systems have only recently attacked this problem. There are two main thrusts towards managing a set of beliefs in the face of uncertain evidence.

Probably the better known of these is that used in expert systems [17]. These methods, based on subjective Bayesian methods, deal with assignment of a numerical weight to beliefs. The more certain a belief is, the higher its weight is. The evidence is examined (typically by a rule-based system) and as more supportive evidence is found, the weight is increased. If conflicting evidence is found, the weight is decreased.

There are a number of advantages and disadvantages this method of coping with uncertainty.* Computers have been built with number-crunching in mind. It is much easier to retain and process information that is represented by a few numbers. When an evaluation is needed, generally a single answer can be given, e.g. "*A* has a weight of 0.9 and so is strongly indicated." If the domain has statistical validity, then this type system is probably indicated. That is, if probabilistic reasoning is justified and the available evidence contributes to the determination of the probabilities, then a numeric system would be appropriate. The mineral geology domain of Prospector [29] has this nature. For instance, the entire surface of the earth presents a large number of possible mineral sites while the data about actual mineral deposits is large enough to make statistical observations.

Other problems with this numeric analysis of uncertainty include the uncertainty of the data and rules in the system. In the Prospector case, a geologist informs the system that the presence of feature F_1 and the absence of feature F_2 give an indication of feature F_3 with a certainty factor of p. There are two problems with this type of rule. One is that the certainty factor is simply an educated guess on the part of the geologist.† If his guess is off, then calculations based on it will be off (but may be tempered by other data). Secondly, the geologist has chosen a certain set of indicators for the application of the rule. He may have neglected to include other indicators for or against the applicability of

*A number of the disadvantages presented here may be overcome by using Shafer-Dempster theory [50]. See especially the evidential reasoning work of Lowrance and Garvey [36].

†However, in other fields, the certainty factor may be based upon statistical studies and would probably be accurate.

this one rule. Even though they may be covered in other rules, this rule may give unwarranted support for a belief. Only full testing of the system will indicate whether it corresponds to the known examples in its domain.

If the system returns a particular weighting for some fact, the actual numeric value is not very accurate. These systems will generally classify the relative belief into one of ten or so classes ranging from strongly indicated through indecisive to strongly contraindicated. However, in doing the internal calculations, actual numbers are used and could possibly lead to instabilities. It would be possible for a slight change in some numbers to lead to quite different conclusions.

Of course such systems are an improvement over a system that does not concern itself with uncertainty—one that considers propositions to be only true or false. If a system does not distinguish between the levels of belief in the knowledge that it knows, it can not properly decide what paths of reasoning to follow. It might spend much of its time wandering on deductive pathways that are based upon the slimmest evidence only to find the evidence cut away later.

On the other hand, a system that has knowledge about its levels of belief should be able to better guide its search for new knowledge. It can choose to avoid reasoning that involves uncertain knowledge in favor of reasoning about more certain knowledge. Its processing would be more likely constructive. Yet it can also postulate that some uncertain knowledge is true (or false) and can consider the consequences. In particular, it could do a *reductio ad absurdum* reasoning and determine that the uncertain knowledge would lead to contradictions.

How does one build such a system? Traditional predicate logic systems do not differentiate between formulas except as to whether they are true or false; there is no in-between. However, this is a reflection of the system employing that logic rather than the logic itself. While a two-valued logic does not allow for half-truths, the system employing that logic may recognize that the path leading to some deduction is too tenuous to give much credence to the deduction. So, although the logic may support particular conclusions, the system can be selective about which deductions to allow to be added back into the system. In this way, paths of reasoning which are more well founded will be followed.

In order to retain completeness, the uncertain paths do not need to be discarded. It would probably be advantageous to have some type of priority system that would take into account the uncertainty of a formula along with other relevant information and would order the formulas according to an expected utility. By basing the priority of a formula on more than just its certainty factor, the system can better choose where its search will go. A portion of the priority may be based upon the *nature* of the uncertainty as well as on the amount of uncertainty. For instance, suppose a belief has gained its uncertainty as a result of being at the end of a long deductive chain, each link of which added some more uncertainty. The system might examine what constituted the reason for the uncertainty measure and choose to devalue the uncertainty as far as the priority was concerned if the belief was important to the system. The reasoning behind this is that it has no particular evidence to contradict the belief (but it is getting farther out on a limb) so, if the belief may be crucial in the reasoning, then it would be reasonable to go ahead and try it. On the other hand, if there were contradictory evidence to the belief, then it would not be as reasonable to use it.

We have not yet talked about how the uncertainty is judged or propagated. If we already are keeping track of the basis for beliefs with a TMS, then it is not much more difficult to keep track of the associated certainties. The first component of a certainty measure would be based upon the certainty of the premises upon which the belief is based. This not only includes the beliefs but also the rules that were used in deducing the belief under question.

For the sake of efficiency, there should be some numeric estimate or ranking of certainty for beliefs. Certainly the results of previous work on combining certainty factors would be useful here. However, by keeping this other information, the system can adjust its certainty factors as knowledge grows. If the certainty of one of the supporting beliefs changes, then the certainty factor can be altered. The acquisition of further information may possibly make moot earlier certainty considerations. For instance, determining that it is, in fact, raining supersedes any earlier computations based upon a forecast of rain. Any certainty factors based on the forecast should be removed from the calculation of the current certainty. With all the information around, this would be possible.

There are still many factors to be considered in working with uncertainty. A different, but interesting and possibly applicable, approach has been taken by Cohen and Grinberg [14]. Their system of using endorsements has a potential for being useful for reasoning under uncertainty.

VI SUMMARY

Lives of individuals and the outcomes of battles and wars depend upon the accuracy and responsiveness of military planning. In earlier times, when battles could take days while troop movements and intelligence operations could take weeks, the length of time it took to plan engagements was immaterial.

As technology has increased, the time dimension of tactics has shrunk. Especially in the Air Force, the enemy can be engaged within minutes of a decision to do so. Likewise, the enemy can attack quickly in response to developments. Both tactical and defensive planning must be done quickly—the side that is faster and better prepared will have the advantage. Still, plans must be accurate. Planning too quickly may cause important information to be overlooked—information that may affect whether the plan will achieve its goal.

Again technology has complicated the planning process as intelligence information can be gathered almost instantaneously. This plethora of information offers planners a better chance at having accurate plans but at the same time challenges them because of its sheer volume. Information from radio communication, radar, and satellite reconnaissance is continuously gathered offering up-to-the-second status updates. The amount of the raw information is immense, even by computer standards. Even after it is digested, the available information can only be selectively used by military planners. Whomever most effectively uses this information will have the best plans.

Because the information arrives so fast, it is possible to use it to assess and possibly modify operations already under way. Determining if any adjustments or alterations of the current plan are advisable suffers from the time-critical deadlines in the execution of the plan. These are the points in the goal at which events are scheduled to occur from which there is no simple backtracking. An example might be a plane's delivering its ordinance. It will take a great deal of time before that plane is available to attack another target.

So how can military planning take advantage of the vast information and yet be responsive to the changing situations in an appropriately short time? Technology has added the complications and can be used to manage them. Computers already process raw information from sensors and output the digested form. Also they are used in situations where human reaction time is too slow.

While people are the best judge of a plan if they understand all the complications and ramifications of it, computers can also aid the planning process. One way is to use them to investigate potential plans. If all the proper criteria have been included in the programming, none of these will be forgotten or overlooked. Computers are ideal database machines so they can easily keep track of supplies and other resources used in the plan to make sure they are available. Their calculating abilities allow determining exactly when and where every entity in a plan should be at any moment. High speed communications between computers provides for instant access to the latest information.

With these capabilities, computers should be able to support decision-making and planning, but currently, for a number of reasons, they do not approach their potential use in this field. One major reason is that understanding of planning and modeling of real world situations are inadequate. These inadequacies involve:

- World models.
- A model of time.
- Understanding of inaccurate information.
- Propagating the effects of information and retracing (backtracking) that propagation if necessary.
- Processing speed, especially as regards deduction and simulation.

In this report we have reviewed these difficulties and developed approaches to solving them.

We presented an approach towards developing world models in Chapter III. The problems of state-based world models were discussed and temporal logic was presented as an alternative means of describing the world and its changes.

This logic is based upon first-order predicate calculus but includes temporal predicates to describe the validity of the formulas with respect to time. In this context we define the concept of a STATE (all that is true at an instant), PROCESSES (that cause the STATE to change over time), EVENTS (a description of a set of STATE changes), and CIRCUMSTANCES (consisting of the initial state and the set of processes that cause change in the state). An EXPECTED HISTORY, based upon a view from some point in time, describes what is believed (at that time) to have been true before then and also what is expected to be true in the future.

In Chapter IV we present that portion of the world model that describes plans. The appendices contain examples of such a model. The CONTEXT of a plan is a description of the world in which the plan is executed. This planned execution is the ACTIVITY while the GOAL, PLAN, STEP, and TASK describe the plan itself. ACTIVITY REPORTS are descriptions of what actually happens in the world when the plan is carried out. By comparing these to the planned ACTIVITY (which contained the original EXPECTED HISTORY, conditions may be found necessitating REPLANNING REPORTS which cause the plans to be modified.

In Chapter V we present a design for a system to do real-time monitoring and replanning. The MONITOR has overall responsibility for the system, interfacing both to the people operating it and data reporting systems. Among the components it oversees are the SIMULATION and DEDUCTION components. We feel that the deduction component is the heart of any system to do execution-time monitoring and replanning. While simulation systems only allow the determination of future conditions from earlier conditions, a deduction system can propagate conditions either forward or backward in time.

We discuss possible types of deduction systems that could be employed. Certain refinements of resolution theorem proving appear to be the most promising for the task. These have the advantage of being able to deal with a large body of facts quickly. Among those we discuss are PROLOG, connection-graph resolution, and theory resolution.

Truth maintenance systems provide for auditing information used in deducing other information. If some "fact" in a database is later found to be incorrect, the system can gracefully recover by recanting just those facts dependent upon the erroneous data.

Reasoning in which some facts that may have been believed at one point are not necessarily believed later are called nonmonotonic. For example, default reasoning, where the default is presumed unless proven otherwise, is nonmonotonic. A truth maintenance system is another example. Any system that deals with the uncertainties of the real world, and especially a replanning system, must have a nonmonotonic deduction capability.

Often we can quantify uncertainty; rather than say a proposition is true or false, we say it has an 80% chance of being true. We feel that including probabilities in the world model, while eventually desirable, is currently premature. There are already enough difficulties in creating and maintaining a world model. However we do discuss some approaches to handling uncertainties, such as expert systems.

An annotated bibliography of all the major articles pertaining to the subject domain is included as an appendix.

We recommend that this line of study be continued. The next step in developing an execution-monitoring and replanning system should be to build a small one along the lines presented here for a limited world. There are many practical problems left and attempting to build a system will expose the most critical ones so they may be studied more closely. By building a limited system, certain obvious difficulties such as execution speed can be avoided until the more theoretical problems are resolved. As the implementation of more realistic solutions progresses, the processing requirements can be studied to determine what, if any, modifications are necessary to the design to allow it to run on real-world models in real time.

Appendix A

Formal Model of Sample Scenario

The following is a partial model of the original plans of the example scenario in Section II-D. This shows how the plans and goals fit together in our formalism.

Context:

Work Period: [T_{begin}, T_{end}]

Environment:

LOC(Target1, Location(Target1))

LOC(Target2, Location(Target2))

LOC($SA8_1$, Location($SA8_1$))

LOC($SA8_2$, Location($SA8_2$))

LOC(Airbase1, Location(Airbase1))

LOC(Airbase2, Location(Airbase2))

LOC is used here to indicate that the locations are known to the planner. This may be used in determining how long it takes to fly from Airbase1 to Target1 for the route.

Distance(Location(Target2), Location($SA8_1$)) < SA8range

Distance(Location(Target2), Location($SA8_2$)) < SA8range

Distance would actually be computed, so this need not be included in any real system but is included here for clarity.

Defense($SA8_1$, SA8)

Defense($SA8_2$, SA8)

All x (Defense(x, SA8) and State(x,Functional) $\Rightarrow$

All y (Distance(Location(y), Location(x)) < $SA8_{range}$ $\Rightarrow$ Protects(x, y)))

This specifies that, if the SA8 sites are working, they protect Target2.

Other specifications may go here.

State Axioms:

Holdsat(Unprotected(Target1), T_{begin})

Holdsat(State($SA8_1$,Functional), T_{begin})

Holdsat(State($SA8_2$,Functional), T_{begin})

Other STATE AXIOMS will go here.

Activity:

A simplified visual explanation of what follows is

$$\begin{tikzcd}
P1 \arrow[r, equal] & P2 \arrow[d, dash] & \\
 & P3 \arrow[r] \arrow[d, dash] & P4 \\
P5 \arrow[r, equal] & P6 , &
\end{tikzcd}$$

where P1 is Plan 1, etc. P1 ==== P2 means that P1 and P2 can be executed in parallel. P3 --> P4 means that P3 precedes P4. Both P3 and P4 are subplans of P2. Similarly, both P5 and P6 are subplans of P3.

P1 is the attack on Target1. P2 is the overall plan to attack Target2. P3 is the plan to knock out the defenses of Target2 while P4 is the actual attack on Target2. P5 and P6 are the attacks on the individual SA8 sites.

The formal structure of the Activity is:

Goals:

G1:	Goal:	Holdsat(State(Target1, Destroyed), T_{end})
	Rationale:	Top-level goal
G2:	Goal:	Holdsat(State(Target2, Destroyed), T_{end})
	Rationale:	Top-level goal

Plans:

P1: This is a straightforward plan in which there is only step: a mission (M1) to destroy Target1.

Method/Schedule:

The procedural network contains only one Step: the mission M1. See Appendix B for the description of a MISSION.

Conditions: The conditions are just those of the STEP:

Holdsat(Unprotected(Target1), M1AttackInterval)

Holdsat(Assigned(M1Aircraft, M1), M1Interval)

Holdsat(TakeoffOK(Airbase1, M1), M1TakeoffInterval)

Holdsover(Clear(M1OutPath), M1OutInterval)

Holdsover(Clear(M1InPath), M1InInterval)

Holdsover(LandingOK(Airbase1, M1), M1LandInterval)

Duration(M1AttackInterval) > MinAttackInterval(M1)

Duration(M1AttackInterval) < MaxAttackInterval(M1)

Duration(M1TakeoffInterval) > MinTakeoffInterval(M1)

The first condition is found to be true because of the STATE AXIOMS and an assignment of time to the variable endpoints of M1AttackInterval. The others will also be found true without causing any actions to be taken. The CONDITIONS HISTORY will include the assumptions that these are based on, with pointers back to this plan.

Effects: The EFFECTS are just those of the STEP:

Holdsover(State(Target1, Destroyed), [M1AttackInterval$_{end}$, T$_{end}$])

This effect will be placed on the CONDITIONS HISTORY with pointers back to the PLAN that caused it. Note that this effect satisfies the GOAL CONDITION for G1.

P2: This is a more complex plan. It is basically similar to P1 but it requires a subplan to satisfy the condition that Target2 be unprotected.

Conditions: Inherited from the ENVIRONMENT and STATE AXIOMS.
Effects: Inherited up from the subplans but must include Holdsat(State(Target1, Destroyed), T$_{end}$).
Method/Schedule: The procedural network contains two nodes: G3 and G4. G4 is a goal to achieve the same condition that G2 has, namely, the destruction of Target2. Because a condition needs to be achieved, the PLAN was not a simple STEP and so a subgoal was necessary. G3 is the goal to achieve that condition, the neutralization of the defenses of Target1.

G4:
Goal: Holdsat(State(Target2, Destroyed), T$_{end}$)
Rationale: Satisfy the condition of G2

P4: This goal has essentially the same plan as P1: a mission (M2) that attacks the target. The difference is that some additional conditions are shared with G5. The attack interval must be within 8 hours of the destruction of the SA8 sites.

Conditions: Inherited from P2 with the addition of the effect of P3.
Effects: Holdsat(State(Target2, Destroyed), [M2AttackInterval$_{end}$, T$_{end}$])

G3:

Goal: Holdsover(Unprotected(Target2), M2AttackInterval)

Rationale: Satisfy a condition for a STEP under P4

P3: The METHOD/SCHEDULE of this plan is a procedural network of two nodes that may (or may not) be executed in parallel. The two nodes represent the attacks on the SA8 sites defending Target 2. These nodes will be Goals G5 and G6.

Conditions: Inherited from P2.

Effects: Inherited up from P5 and P6. By noting that the only known defenses of Target2 are destroyed over the required interval, the EFFECTS will make it possible to prove that the goal has been satisfied. (This requires either confirming that these are indeed the only defenses or using the closed-world assumption.)

G5:

Goal: Holdsover(State($SA8_1$, Destroyed), M2AttackInterval)

Rationale: Satisfy the goal condition for G3

P5: This plan is just a step with a mission (M3) that attacks $SA8_1$. The timing of the mission must be such that $M3AttackInterval_{end} < M2AttackInterval_{start}$ and $M3AttackInterval_{end} + 8$ hours $> M2AttackInterval_{end}$. Otherwise P5 is similar to P1.

Conditions: Inherited from P3

Effects: Holdsover(State($SA8_1$, Destroyed), [$M3AttackInterval_{end}$, $M3AttackInterval_{end}$ + 8 Hours])

From this EFFECT and the timing requirements, the GOAL of G5 can be shown to have been satisfied.

G6: is similar to G5, but is a mission (M4) against $SA8_2$.

Appendix B

Formal Model of an Air Interdiction Mission

Below we present an indication of how air interdiction missions can be modeled within our formalism.

Mission(Base, Aircraft, Target, OutPath, InPath, LaunchInterval,
OutInterval, AttackInterval, InInterval, LandInterval)

The input specification must specify the upper bound of at least one interval and the lower bound of at least one interval. All the other intervals can be computed from the other parameters. It is an error if the specified times are incompatible.

Method:
- Launch(Base, Aircraft);
- Flyout(Aircraft, OutPath);
- Engage(Aircraft, Target);
- Flyin(Aircraft, InPath);
- Land(Base, Aircraft).

Schedule:
- T_{start} = $\text{LaunchInterval}_{start}$
- D_{min} = $\text{LandInterval}_{start}$ - $\text{LaunchInterval}_{end}$
- D_{exp} equals the sum of D_{exp} of all the subtasks.
- D_{max} = $\text{LandInterval}_{end}$ - $\text{LaunchInterval}_{start}$

Instruments:
- Aircraft

Conditions:
- HOLDSAT(Assigned(Aircraft, ThisMission), TSTART);
- HOLDSAT(Prepared(Aircraft), TSTART);
- HOLDSAT(Open(Base, Runways), TSTART);
- HOLDSOVER(Clear(OutPath), OutInterval),
- HOLDSOVER(Clear(InPath), InInterval),
- HOLDSOVER(Undefended(Target), AttackInterval),
- HOLDSOVER(Open(Base, Runways), LandInterval).

Effects:

HOLDSOVER(Busy(Aircraft),[$\text{LaunchInterval}_{start}$, $\text{LandInterval}_{end}$]);

HOLDSDURING(State(Target, Destroyed), AfterAttackInterval),
  where AfterAttackInterval is [$\text{AttackInterval}_{end}$, T_{end}];

Appendix C

Listing of Process and Event Example

Here is an annotated listing of the program that processes constraints on the time intervals for PROCESSES and EVENTS.

First, we create a process called TASK1 that describes the activity of running for five to ten minutes.

```
_(CREATE-PROCESS 'TASK1 '(5 10) 'RUNNING)

_(SHOW TASK1]
   PROCESS    PROCESS1
      NAME:     TASK1
       DUR:     (5 10)
      DESC:     RUNNING
      BEGIN:
```

This describes the beginning of the process. The beginning of a process is an EVENT, in this case EVENT2.

```
      EVENT    EVENT2
       NAME:     EVENT2
       DESC:     (BEGIN PROCESS1)
       TIME:
```

An EVENT takes place at an INSTANT. Here we describe that INSTANT. TA is the earliest time at which the INSTANT might really have occurred, while TB is the latest time. TAC lists the constraints on TA while TBC lists the constraints on TB. IC lists the constraints on the INSTANT itself.

```
         INSTANT    INST2
            DESC:     (TIME EVENT2)
              TA:     MINTIME
              TB:     MAXTIME
             TAC:     (*LE* (TA INST2) (*MINUS* (TA INST3) 5))
                      (*GE* (TA INST2) (*MINUS* (TA INST3) 10))
             TBC:     (*GE* (TB INST2) (*MINUS* (TB INST3) 10))
                      (*LE* (TB INST2) (*MINUS* (TB INST3) 5))
              IC:     (PRECEDES (INST3 (5 10)))
   END:

      EVENT    EVENT3
       NAME:     EVENT3
   DESC:    (END PROCESS1)
   TIME:

    INSTANT    INST3
       DESC:    (TIME EVENT3)
         TA:    MINTIME
         TB:    MAXTIME
        TAC:    (*GE* (TA INST3) (*PLUS* (TA INST2) 5))
                (*LE* (TA INST3) (*PLUS* (TA INST2) 10))
        TBC:    (*LE* (TB INST3) (*PLUS* (TB INST2) 10))
                (*GE* (TB INST3) (*PLUS* (TB INST2) 5))
         IC:    (SUCCEEDS (INST2 (5 10)))
```

Next we create another process of resting for a few (4 to 8) minutes. The result is directly analogous to TASK1 and will not be shown.

```
_(CREATE-PROCESS 'TASK2 '(4 8) 'RESTING)
```

Now we assert that resting (in a not too strenuous exercise plan) comes a few minutes after running.

```
_(ASSERT (AFTER TASK2 TASK1 '(3 6]
T
```

The T indicates this can be asserted consistently, with the following results:

```
_(SHOW TASK1)
   PROCESS    PROCESS1
      NAME:    TASK1
       DUR:    (5 10)
      DESC:    RUNNING
     BEGIN:

        EVENT    EVENT2
         NAME:    EVENT2
         DESC:    (BEGIN PROCESS1)
         TIME:

          INSTANT    INST2
             DESC:    (TIME EVENT2)
               TA:    MINTIME
               TB:    MAXTIME
              TAC:    (*LE* (TA INST2) (*MINUS* (TA INST3) 5))
                      (*GE* (TA INST2) (*MINUS* (TA INST3) 10))
              TBC:    (*GE* (TB INST2) (*MINUS* (TB INST3) 10))
                      (*LE* (TB INST2) (*MINUS* (TB INST3) 5))
               IC:    (PRECEDES (INST3 (5 10)))
     END:

        EVENT    EVENT3
         NAME:    EVENT3
         DESC:    (END PROCESS1)
         TIME:
```

```
        INSTANT    INST3
           DESC:     (TIME EVENT3)
             TA:     MINTIME
             TB:     MAXTIME
            TAC:     (*GE* (TA INST3) (*PLUS* (TA INST2) 5))
                     (*LE* (TA INST3) (*PLUS* (TA INST2) 10))
                     (*LE* (TA INST3) (*MINUS* (TA INST4) 3))
                     (*GE* (TA INST3) (*MINUS* (TA INST4) 6))
            TBC:     (*LE* (TB INST3) (*PLUS* (TB INST2) 10))
                     (*GE* (TB INST3) (*PLUS* (TB INST2) 5))
                     (*GE* (TB INST3) (*MINUS* (TB INST4) 6))
                     (*LE* (TB INST3) (*MINUS* (TB INST4) 3))
             IC:     (SUCCEEDS (INST2 (5 10)))
                     (PRECEDES (INST4 (3 6)))


_(SHOW TASK2)
  PROCESS    PROCESS2
     NAME:    TASK2
      DUR:    (4 8)
     DESC:    RESTING
     BEGIN:

        EVENT    EVENT4
         NAME:    EVENT4
         DESC:    (BEGIN PROCESS2)
         TIME:

          INSTANT    INST4
             DESC:     (TIME EVENT4)
               TA:     MINTIME
               TB:     MAXTIME
              TAC:     (*LE* (TA INST4) (*MINUS* (TA INST5) 4))
                       (*GE* (TA INST4) (*MINUS* (TA INST5) 8))
                       (*GE* (TA INST4) (*PLUS* (TA INST3) 3))
                       (*LE* (TA INST4) (*PLUS* (TA INST3) 6))
              TBC:     (*GE* (TB INST4) (*MINUS* (TB INST5) 8))
                       (*LE* (TB INST4) (*MINUS* (TB INST5) 4))
                       (*LE* (TB INST4) (*PLUS* (TB INST3) 6))
                       (*GE* (TB INST4) (*PLUS* (TB INST3) 3))
               IC:     (PRECEDES (INST5 (4 8)))
                       (SUCCEEDS (INST3 (3 6)))
     END:

        EVENT    EVENT5
         NAME:    EVENT5
         DESC:    (END PROCESS2)
         TIME:

          INSTANT    INST5
             DESC:     (TIME EVENT5)
               TA:     MINTIME
               TB:     MAXTIME
              TAC:     (*GE* (TA INST5) (*PLUS* (TA INST4) 4))
                       (*LE* (TA INST5) (*PLUS* (TA INST4) 8))
              TBC:     (*LE* (TB INST5) (*PLUS* (TB INST4) 8))
```

```
                    (*GE* (TB INST5) (*PLUS* (TB INST4) 4))
             IC:    (SUCCEEDS (INST4 (4 8)))
```

Note that new constraints have been placed between aspects of the beginning time of TASK2 and the ending time of TASK1, but that the respective values of TA and TB remain consistent and unchanged.

Now let us suppose we see someone running at about time 13.

```
_(CREATE-EVENT 'SEE1 'RUNNING)
EVENT6
_(SHOW SEE1)

        EVENT    EVENT6
         NAME:    SEE1
         DESC:    RUNNING
         TIME:

          INSTANT    INST6
             DESC:    (TIME EVENT6)
               TA:    MINTIME
               TB:    MAXTIME
              TAC:
              TBC:
               IC:

_(ASSERT (AT SEE1 '(12 14]

_(SHOW SEE1)

        EVENT    EVENT6
         NAME:    SEE1
         DESC:    RUNNING
         TIME:

          INSTANT    INST6
             DESC:    (TIME EVENT6)
               TA:    12
               TB:    14
              TAC:    (*GE* (TA INST6) 12)
              TBC:    (*LE* (TB INST6) 14)
               IC:
```

Now we assert that this is part of that person's exercise plan; specifically, that it occurs during TASK1 of our simple plan.

```
_(ASSERT (DURING SEE1 TASK1))
T
```

We find that this can be done consistently, of course. This binds instants of our plan to actual intervals, as constraints are propagated. The results are shown below; new and altered elements are marked by asterisks.

```
_(SHOW TASK1)
   PROCESS    PROCESS1
      NAME:    TASK1
       DUR:    (5 10)
      DESC:    RUNNING
      BEGIN:

         EVENT    EVENT2
          NAME:    EVENT2
          DESC:    (BEGIN PROCESS1)
          TIME:

            INSTANT    INST2
               DESC:    (TIME EVENT2)
                 TA:    2 *
                 TB:    14 *
                TAC:    (*LE* (TA INST2) (*MINUS* (TA INST3) 5))
                        (*GE* (TA INST2) (*MINUS* (TA INST3) 10))
                        (*LE* (TA INST2) (TA INST6)) *
                TBC:    (*GE* (TB INST2) (*MINUS* (TB INST3) 10))
                        (*LE* (TB INST2) (*MINUS* (TB INST3) 5))
                        (*LE* (TB INST2) (TB INST6)) *
                 IC:    (PRECEDES (INST3 (5 10)) (INST6 (0 10)))
      END:

         EVENT    EVENT3
          NAME:    EVENT3
          DESC:    (END PROCESS1)
          TIME:

            INSTANT    INST3
               DESC:    (TIME EVENT3)
                 TA:    12 *
                 TB:    24 *
                TAC:    (*GE* (TA INST3) (*PLUS* (TA INST2) 5))
                        (*LE* (TA INST3) (*PLUS* (TA INST2) 10))
                        (*LE* (TA INST3) (*MINUS* (TA INST4) 3))
                        ~~(*GE* (TA INST3) (*MINUS* (TA INST4) 6))~~
                        (*GE* (TA INST3) (TA INST6)) *
                TBC:    (*LE* (TB INST3) (*PLUS* (TB INST2) 10))
                        (*GE* (TB INST3) (*PLUS* (TB INST2) 5))
                        (*GE* (TB INST3) (*MINUS* (TB INST4) 6))
                        (*LE* (TB INST3) (*MINUS* (TB INST4) 3))
                        (*GE* (TB INST3) (TB INST6)) *
                 IC:    (SUCCEEDS (INST2 (5 10)) (INST6 (0 10)))
                        (PRECEDES (INST4 (3 6)))


_(SHOW TASK2)
   PROCESS    PROCESS2
      NAME:    TASK2
       DUR:    (4 8)
      DESC:    RESTING
      BEGIN:
```

```
   EVENT    EVENT4
    NAME:    EVENT4
    DESC:    (BEGIN PROCESS2)
    TIME:

      INSTANT    INST4
         DESC:    (TIME EVENT4)
           TA:    15 *
           TB:    30 *
          TAC:    (*LE* (TA INST4) (*MINUS* (TA INST5) 4))
                  (*GE* (TA INST4) (*MINUS* (TA INST5) 8))
                  (*GE* (TA INST4) (*PLUS* (TA INST3) 3))
                  (*LE* (TA INST4) (*PLUS* (TA INST3) 6))
          TBC:    (*GE* (TB INST4) (*MINUS* (TB INST5) 8))
                  (*LE* (TB INST4) (*MINUS* (TB INST5) 4))
                  (*LE* (TB INST4) (*PLUS* (TB INST3) 6))
                  (*GE* (TB INST4) (*PLUS* (TB INST3) 3))
           IC:    (PRECEDES (INST5 (4 8)))
                  (SUCCEEDS (INST3 (3 6)))
END:

   EVENT    EVENT5
    NAME:    EVENT5
    DESC:    (END PROCESS2)
    TIME:

      INSTANT    INST5
         DESC:    (TIME EVENT5)
           TA:    19 *
           TB:    38 *
          TAC:    (*GE* (TA INST5) (*PLUS* (TA INST4) 4))
                  (*LE* (TA INST5) (*PLUS* (TA INST4) 8))
          TBC:    (*LE* (TB INST5) (*PLUS* (TB INST4) 8))
                  (*GE* (TB INST5) (*PLUS* (TB INST4) 4))
           IC:    (SUCCEEDS (INST4 (4 8)))


_(SHOW SEE1)

   EVENT    EVENT6
    NAME:    SEE1
    DESC:    RUNNING
    TIME:

      INSTANT    INST6
         DESC:    (TIME EVENT6)
           TA:    12
           TB:    14
          TAC:    (*GE* (TA INST6) 12)
                  (*GE* (TA INST6) (TA INST2)) *
                  (*LE* (TA INST6) (TA INST3)) *
          TBC:    (*LE* (TB INST6) 14)
                  (*LE* (TB INST6) (TB INST3)) *
                  (*GE* (TB INST6) (TB INST2)) *
           IC:    (PRECEDES (INST3 (0 10)))
                  (SUCCEEDS (INST2 (0 10)))
```

Now suppose we see that person resting at about time 20 and we assert this as being part of his exercise activity.

```
_(CREATE-EVENT 'SEE2 'RESTING)
EVENT7
_(ASSERT (AT SEE2 '(18 22]
T
_(ASSERT (DURING SEE2 TASK2]
T
```

We find this is possible, and that it further constrains the intervals. The results are shown on the following pages, with these changes also marked by asterisks.

```
_(SHOW TASK1)
   PROCESS    PROCESS1
      NAME:    TASK1
       DUR:    (5 10)
      DESC:    RUNNING
      BEGIN:

         EVENT    EVENT2
          NAME:    EVENT2
          DESC:    (BEGIN PROCESS1)
          TIME:

           INSTANT    INST2
              DESC:    (TIME EVENT2)
                TA:    2
                TB:    14
               TAC:    (*LE* (TA INST2) (*MINUS* (TA INST3) 5))
                       (*GE* (TA INST2) (*MINUS* (TA INST3) 10))
                       (*LE* (TA INST2) (TA INST6))
               TBC:    (*GE* (TB INST2) (*MINUS* (TB INST3) 10))
                       (*LE* (TB INST2) (*MINUS* (TB INST3) 5))
                       (*LE* (TB INST2) (TB INST6))
                IC:    (PRECEDES (INST3 (5 10)) (INST6 (0 10)))
      END:

         EVENT    EVENT3
          NAME:    EVENT3
          DESC:    (END PROCESS1)
          TIME:

           INSTANT    INST3
              DESC:    (TIME EVENT3)
                TA:    12
                TB:    19 *
               TAC:    (*GE* (TA INST3) (*PLUS* (TA INST2) 5))
                       (*LE* (TA INST3) (*PLUS* (TA INST2) 10))
                       (*LE* (TA INST3) (*MINUS* (TA INST4) 3))
                       (*GE* (TA INST3) (*MINUS* (TA INST4) 6))
                       (*GE* (TA INST3) (TA INST6))
               TBC:    (*LE* (TB INST3) (*PLUS* (TB INST2) 10))
```

```
                 (*GE* (TB INST3) (*PLUS* (TB INST2) 5))
                 (*GE* (TB INST3) (*MINUS* (TB INST4) 6))
                 (*LE* (TB INST3) (*MINUS* (TB INST4) 3))
                 (*GE* (TB INST3) (TB INST6))
          IC:    (SUCCEEDS (INST2 (5 10)) (INST6 (0 10)))
                 (PRECEDES (INST4 (3 6)))


_(SHOW TASK2)
  PROCESS    PROCESS2
     NAME:    TASK2
      DUR:    (4 8)
     DESC:    RESTING
     BEGIN:

        EVENT     EVENT4
         NAME:     EVENT4
         DESC:     (BEGIN PROCESS2)
         TIME:

          INSTANT    INST4
             DESC:    (TIME EVENT4)
               TA:    15
               TB:    22 *
              TAC:    (*LE* (TA INST4) (*MINUS* (TA INST5) 4))
                      (*GE* (TA INST4) (*MINUS* (TA INST5) 8))
                      (*GE* (TA INST4) (*PLUS* (TA INST3) 3))
                      (*LE* (TA INST4) (*PLUS* (TA INST3) 6))
                      (*LE* (TA INST4) (TA INST7)) *
              TBC:    (*GE* (TB INST4) (*MINUS* (TB INST5) 8))
                      (*LE* (TB INST4) (*MINUS* (TB INST5) 4))
                      (*LE* (TB INST4) (*PLUS* (TB INST3) 6))
                      (*GE* (TB INST4) (*PLUS* (TB INST3) 3))
                      (*LE* (TB INST4) (TB INST7)) *
               IC:    (PRECEDES (INST5 (4 8)) (INST7 (0 8)))
                      (SUCCEEDS (INST3 (3 6)))
     END:

        EVENT     EVENT5
         NAME:     EVENT5
         DESC:     (END PROCESS2)
         TIME:

          INSTANT    INST5
             DESC:    (TIME EVENT5)
               TA:    19
               TB:    30 *
              TAC:    (*GE* (TA INST5) (*PLUS* (TA INST4) 4))
                      (*LE* (TA INST5) (*PLUS* (TA INST4) 8))
                      (*GE* (TA INST5) (TA INST7)) *
              TBC:    (*LE* (TB INST5) (*PLUS* (TB INST4) 8))
                      (*GE* (TB INST5) (*PLUS* (TB INST4) 4))
                      (*GE* (TB INST5) (TB INST7)) *
               IC:    (SUCCEEDS (INST4 (4 8)) (INST7 (0 8)))
```

Now we see resting at about time 10. When we try to place this fact during TASK2 of the exercise plan, an inconsistency is discovered. The reason for this is that 15 is the earliest the process can start if it is to be consistent with prior assertions.

```
_(CREATE-EVENT 'SEE3 'RESTING)
EVENT8
_(ASSERT (AT SEE3 '(9 11)))
T
_(ASSERT (DURING SEE3 TASK2))



CAN'T ASSERT THE STATEMENT (DURING SEE3 TASK2)
AS (*GE* (TA INST8) (TA INST4))
CAUSES A CONTRADICTION AT INSTANT INST8
NIL


_(S OW INST8)

           INSTANT    INST8
              DESC:    (TIME EVENT8)
                TA:    15
                TB:    11
               TAC:    (*GE* (TA INST8) 9)
               TBC:    (*LE* (TB INST8) 11)
                IC:    (PRECEDES (INST6 (0 8)))
                       (SUCCEEDS (INST4 (0 8)))
```

Appendix D

An Annotated Bibliography

1. Allen, J. F., "A General Model of Action and Time," Technical Report TR-97, Department of Computer Science, University of Rochester, Rochester, New York (November 1981).

   Discusses a temporal logic based on intervals of time, predicates over intervals (e.g., BEFORE) and the predicate HOLDS for conditions at times. Presents a set of axioms to serve as a basis for reasoning about time.

2. Allen, J F., "An Interval-Based Representation of Temporal Knowledge," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 221-226, University of British Columbia, Vancouver, B. C., Canada (August 1981).

   Introduces a hierarchical, interval-based approach to representing time and maintaining current discourse time.

3. Appelt, D. E., "A Planner for Reasoning about Knowledge and Action," *Proceedings of the National Conference on Artificial Intelligence*, pp. 131-133, Stanford University, Stanford, California (August 1980).

   Describes his KAMP planning system, based upon the NOAH system [47], but using the possible-worlds-semantics approach to representing knowledge about belief.

4. Bledsoe, W. W. and W. M. Tyson, "Typing and Proof by Cases in Program Verification," in *Machine Intelligence 8* (Ellis Horwood Limited, Chichester, England, 1977).

   Describes procedures added to a natural deduction theorem prover to handle inequalities and equalities and proof by cases.

5. Borning, A., "The Programming Language Aspects of ThingLab, A Constraint-Oriented Simulation Laboratory," *ACM Trans. on Programming Languages and Systems*, Vol. 3, No. 4, pp. 353-387 (October 1981).

   Discusses an object-oriented, constraint propagation approach to simulation.

6. Bortels, W. H., "The Mission Effectiveness Program: An Analyst's View," *Proc. 1980 Winter Simulation Conference*, pp. 41-49 (1980).

An adaptable simulation system to assist U.S Navy personnel in evaluating changes in mission effectiveness because of new or reallocated combat units (e.g., aircraft, radars).

7. Bresina, J., "An Interactive Planner that Creates a Structured, Annotated Trace of its Operation," Technical Report CBM-TR-123, Rutgers University, New Brunswick, New Jersey (December 1981).

Describes an interactive planning system, PLANX10, that operates with an incomplete knowledge base and can generate partial plans.

8. Brown, D. R., et al, "R&D Plan for Army Applications of AI/Robotics," Final Report, Contract DAAK70-81-C-0250, SRI Project 3736, SRI International, Menlo Park, California (May 1982).

Report to the Army to help shape the research and development plan for applications of artificial intelligence and robotics in combat and combat support. Of special interest is the discussion of a Brigade Mission Planning Aid.

9. Carbonell, J. G., "The Counterplanning Process: Reasoning Under Adversity," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 124-130, Tokyo, Japan (August 1979).

Describes a strategy-based model of planning for dynamic plan adaptation and replanning in obstructive and constructive counterplanning situations.

10, Chang, C. L., "DEDUCE: A Deductive Query Language For Relational Data Bases," in *Pattern Recognition and Artificial Intelligence* (Academic Press, New York, 1977).

Discusses the use of general intensional rules to reduce complex queries to specific data base requests, whose results must be combined logically to generate appropriate responses.

11. Cheeseman, P., "A Representation of Time for Planning," SRI Artificial Intelligence Center Technical Note (forthcoming), SRI International, Menlo Park, California (1982).

Describes a proposed system in PROLOG for representing time in a planning system. It is able to handle continuously changing variables and causal chains.

12. Clifford, J., and D. S. Warren, "Formal Semantics for Time in Databases," Technical Report TR #81/025, Department of Computer Science, SUNY at Stony Brook, Stony Brook, New York (November 1981).

Describes a model for dynamic data bases that incorporates the intension and extension concepts of Montague. This is similar to the our suggested approach, as it has a sequence of time-stamped data base instances with means for deducing values at unstamped times.

13. Clocksin, W. F. and C. S. Mellish, *Programming in Prolog*, (Springer-Verlag, New York, 1981).

Describes both programming in Prolog and the nature of logic programming.

14. Cohen, P. R. and M. R. Grinberg, "A Theory of Heuristic Reasoning about Uncertainty," *AI Magazine*, Vol. 4, No. 2, pp. 17-24 (1983).

This is a clear, but relatively shallow, introduction to a theory of reasoning about uncertainty based on a representation of states of certainty called endorsements. This system is used in their portfolio management expert system, FOLIO. The basic idea is similar to a TMS [16] but differentiates among types of support. The more support (endorsements) for a conclusion, the more confidence the system has that it is right. When the justifications for a conclusion are stronger, the more weight it has when endorsing other conclusions.

15. Corkill, D. D., "Hierarchical Planning in a Distributed Environment," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 168-175, Tokyo, Japan (August 1979).

Describes a generalization of NOAH [47] to a system with multiple planning centers.

16. Doyle, J., "A Truth Maintenance System," *Artificial Intelligence*, Vol. 12, No. 3, pp. 231-272 (1979).

Presents a reasoning system that keeps track of its beliefs and the reasons for those beliefs. The system is capable of revising its beliefs and properly propagating the changes as necessary. Since every belief in the system has an associated list of support (justifications), the removal of a belief from the system results in the removal of the support of some conclusions based upon that belief.

17. Duda, R. O., P. E. Hart, and N. Nilsson, "Subjective Bayesian Methods for Rule-Based Inference Systems," SRI Artificial Intelligence Center Technical Note 124, SRI International, Menlo Park, California (1976).

18. Engleman, C., C. H. Berg, and M. Bischoff, "KNOBS: An Experimental Knowledge Based Tactical Air Mission Planning System and a Rule Based Aircraft Identification Simulation Facility," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 247-249, Tokyo, Japan (August 1979).

Discusses a planning system using frames and a rule based production system. Although the early system was not interactive, it later became so.

19. Engleman, C., E. A. Scarl, and C. H. Berg, "Interactive Frame Instantiation," *Proceedings of the National Conference on Artificial Intelligence*, pp. 184-186, Stanford University, Stanford, California (August 1980).

A later version of KNOBS [18] in which the user of the system helps to supply or control constraint verification. The constraints are the slots within the frames that represent such entities as interdiction missions.

20. Faletti, J., "PANDORA - A Program for Doing Commonsense Planning in Complex Situations," *Proceedings of the National Conference on Artificial Intelligence*, pp. 185-188, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

A planning program that creates plans in two commonsense domains, (1) everyday situations and (2) an operating-system consultant, using hierarchical planning and meta-planning. The system interleaves creation, simulation, and revision of plans. It simulates the plan at the top level and puts the actions and effects of the plan into the future data base. Inferencing is then done to determine whether there are any problems associated with the actions. If so, the plans are modified to avoid the problems.

21. Faught, W. S., P. Klahr, and G. R. Martins, "An Artificial Intelligence Approach to Large-Scale Simulation," *Proc. 1980 Summer Simulation Conference*, Seattle, Washington (August 1980).

Discusses significant aspects of the ROSS simulation system, which attempts to improve the adaptability and comprehensibility of tactical air simulations. The system is further described in reference [37].

22. Fikes, R. E., and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2, No. 3/4, pp. 189-208 (1971).

Describes an early planning system that served as a model for many subsequent planning systems.

23. Fikes, R. E., "Monitored Execution of Robot Plans Produced by STRIPS," *Proceedings IFIP Congress 1971*, Ljubljana, Yugoslavia (1971).

PLANEX1, a very early system based on STRIPS [22], considered execution of plans in a real environment. Before each step of the plan, all the goals are checked backward from the final goal until one is found that is true. The step is then taken that will move from that goal state to the next and the process is repeated. If the

step fails to achieve its goal, this mechanism ensures that it will be repeated.

24. Fikes, R. E., P. E. Hart, and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence*, Vol. 3, No. 4, pp. 251-288 (1972).

A further description of the PLANEX system [23].

25. Fox, M. S., "The Intelligent Management System: An Overview," Technical Report CMU-RI-TR-81-4, Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1981).

Discusses the role of simulation in production control decision-making. Their system is object-oriented, discrete-event, and closely tied to a windowed-graphics interface.

26. Gaines,R. S., W. E. Naslund, and R. Strauch, "Combat Operations Decisionmaking in Tactical Air Command and Control," Rand Note N-1633-AF, The Rand Corporation, Santa Monica, California (December 1980).

Discusses current procedures in Combat Operations of a TACC. Notes need for information systems maintaining up-to-date plan status and plan interrelationships.

27. Gallaire, H., and J. Minker, *Logic And Databases*, (Plenum Press, New York, 1978).

An important book on the general topic of the relationship between data bases and logic.

28. Georgeff, M. and U. Bonollo, "Procedural Expert Systems," *Proceedings of the International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany (August 1983).

Describes a system of adding procedural information into expert systems. Can be used for planning.

29. Hart, P. E., "Prospector - A Computer-Based Consultation System for Mineral Exploration," *International Association for Mathematical Geology*, Vol. 10, No. 5-6 (1977).

30. Hayes, P. J., "The Naive Physics Manifesto," in *Expert Systems in the Micro-electronic Age*, D. Michie (ed.), pp. 242-270 (Edinburgh University Press, Edinburgh, 1979).

Discusses issues of qualitative, yet formal, models of the real world that people seem to use in reasoning about expected events and explaining past events. Time models play a fundamental role in this reasoning.

31. Hayes-Roth, B., et al, "Modeling Planning as an Incremental, Opportunistic Process," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 375-383, Tokyo, Japan (August 1979).

Describes a planning model for an errand-running domain. The main purpose of the system is to test the sufficiency of the planning model as a psycholcgical theory. The system itself is similar to Hearsay-II.

32. Hendrix, G. G., "Modeling Simultaneous Actions and Continuous Processes," *Artificial Intelligence*, Vol. 4, No. 3/4, pp. 145-180 (1973).

An early paper that indicated the need for event-based simulation in the attempt to model real-world planning.

33. Klahr, P., D. McArthur, and S. Narain, "SWIRL: An Object-Oriented Air Battle Simulator," *Proceedings of the National Conference on Artificial Intelligence*, pp. 331-334, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

This paper describes a simulation system written in ROSS. The objects in the system pass messages back and forth. The behavior of an object is evidenced by the sets of messages it sends. This simulation is fairly coarse-grained, allowing most details to be ignored and minimizing the execution time.

34. Konolige, K., "A First-Order Formalization of Knowledge and Action for a Multiagent Planning System," SRI Artificial Intelligence Center Technical Note 232, SRI International, Menlo Park, California (December 1980).

Discusses a logic of belief and knowledge necessary for planning in multiple-agent contexts. Each agent has a data base of believed propositions, including beliefs as to the contents of other agents' data bases.

35. Konolige, K. and N. J. Nilsson, "Multiple-Agent Planning Systems," *Proceedings of the National Conference on Artificial Intelligence*, pp. 138-141, Stanford University, Stanford, California (August 1980).

Analyzes problems of reasoning about other cooperative agents, what they believe, and what they may do.

36. Lowrance, J. D., and T. D. Garvey, "Evidential Reasoning: An Approach to the Simulation of a Weapons Operation Center," Technical Report SRI International, Menlo Park, California (1983).

Describes evidential reasoning, properly incorporating uncertain and incomplete evidence, based upon a Shafer-Dempster approach.

37. McArthur, D., and P. Klahr, "The ROSS Language Manual," Rand Note N-1854-AF, The Rand Corporation, Santa Monica, California (1982).

38. McArthur, D., R. Steeb, and S. Cammarata, "A Framework for Distributed Problem Solving," *Proceedings of the National Conference on Artificial Intelligence*, pp. 181-184, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

Each aircraft in the domain of air traffic control is considered an agent that must cooperate with others to achieve a conflict-free plan. Each can gather and distribute information, create, evaluate, fix, and execute plans. "...in situations where a given agent is not the sole cause of change,and therefore where not all important consequences of a planned action can be foreseen at the time of planning, it is essential that the agent be able to effectively interleave information gathering, planning and execution tasks." Each agent in this system notices problems in plans by comparing its plans with the stated intentions (plans) of the other agents and by reviewing new information for consistency with beliefs about the others' plans.

39. McCarthy, J., "Circumscription--A Form of Non-Monotonic Reasoning," *Artificial Intelligence*, Vol. 13, pp. 27-39 (1980).

Describes circumscription which is just the assumption that everything that is known to have a particular property is the entire set of objects having that property. Also discusses the qualification problem: How can you so completely specify all the information so that there are not some qualifications that you have not covered. For example, when you decide that a boat may be rowed to an island, did you specify that the boat wouldn't sink, that the water is not too shallow or rough, that there are oars, etc. Circumscription allows you to bypass this problem by specifying that nothing that was not considered is relevant.

40. McDermott, D. and J. Doyle, "Non-Monotonic Logic I," *Artificial Intelligence*, Vol. 13, pp. 41-72 (1980).

Most logical systems presume that new information adds to and does not contradict earlier information. Non-monotonic logical systems do not have this property: new axioms can invalidate old theorems. This paper presents a reasonably thorough introduction to such a logic and its relation to other logics. Non-monotonic logic is critical for systems in which beliefs may need to be changed. This paper is related to Doyle's TMS paper [16].

41. McDermott, D., "A Temporal Logic for Reasoning About Processes and Plans," *Cognitive Science*, Vol. 6, No. 2, pp. 101-156 (1982).

Discusses problems that arise when one tries to base temporal reasoning purely on conditions true at points in time. Gives a set of axioms for reasoning in this model of time.

42. McDermott, D., "Nonmonotonic Logic II: Nonmonotonic Modal Theories," *J. ACM*, Vol. 29, No. 1, pp. 33-57 (1982).

A follow-up to the first Nonmonotonic Logic paper [40], this paper contains more technical presentations of the logic along with a discussion of its shortcomings.

43. Reiter, R., "Deductive Question-Answering on Relational Data Bases," Technical Report 77-15, Department of Computer Science, University of British Columbia, Vancouver, B. C., Canada (October 1977).

Similar to [10]; discusses the interface between a relational data base and logical statements whose truths are to be determined relative to that data base.

44. Rosenschein, S. J., "Plan Synthesis: A Logical Perspective," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 331-337, University of British Columbia, Vancouver, B. C., Canada (August 1981).

Discusses a representation for planning as deductive theorem-proving in a propositional dynamic logic. The system allows for the straightforward statement and solution of disjunctive goals.

45. Rosenschein, J. S., "Synchronization of Multi-Agent Plans," *Proceedings of the National Conference on Artificial Intelligence*, pp. 115-119, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

Describes a planning system that creates plans for multiple agents to carry out. The design is concerned with the scheduling and communication necessary for the agents' actions to occur in the correct order.

46. Sacerdoti, E. D., "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence*, Vol. 5, No. 2, pp. 115-136 (1974).

Describes ABSTRIPS, which is based on STRIPS [22] but solves goals hierarchically.

47. Sacerdoti, E. D., *A Structure for Plans and Behavior*, (Elsevier North-Holland, New York, 1977).

Describes NOAH (Nets Of Action Hierarchies), a planning system upon which many other systems are based. NOAH encodes the plan as a procedural net upon which it operates. Subgoals are generated in parallel, but, if a critic later discovers that they cannot be executed in parallel, NOAH linearizes the plan. NOAH interacts with the user to carry out its plan. If the user reports that the next plan step is not possible, NOAH tries to find where the plan has failed by asking the user to confirm that each plan step has been carried out properly.

48. Sacerdoti, E. D., "Problem Solving Tactics," *Proceedings of the International Joint Conference on Artificial Intelligence,* pp. 1077-1085, Tokyo, Japan (August 1979).

An overview of a number of basic strategies for planning systems and a variety of tactics for improving their efficiency.

49. Schwartz, R. L., P. M. Melliar-Smith, and F. H. Vogt, "An Interval Logic for Higher-Level Temporal Reasoning: Language Definition and Examples," CSL 138, SRI International, Menlo Park, California (February 1983).

Describes a temporal logic based on intervals that was developed for program verification. The ideas are interesting because of the parallels of program execution as compared to plan execution.

50. Shafer, G., *A Mathematical Theory of Evidence,* (Princeton University Press, Princeton, New Jersey, 1976).

51. Sridharan, N. S., and J. L. Bresina, "Plan Formation in Large, Realistic Domains," Technical Report CBM-TR-127, Rutgers University, New Brunswick, New York (March 1982).

A further description of PLANX10 [7].

52. Steeb, R. and S. C. Johnson, "A Computer-Based Interactive System for Group Decisionmaking," *IEEE Trans. on Systems, Man, and Cyber.,* Vol. 11, No. 8, pp. 544-552 (August 1981).

Discusses an interactive technique for the determination and evaluation of options.

53. Stickel, M. E., "A Nonclausal Connection-Graph Resolution Theorem-Proving Program," *Proceedings of the National Conference on Artificial Intelligence,* pp. 229-233, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

Describes one of the most convenient and most powerful of the current theorem proving programs. The present implementation is designed for a system that will have larger amounts of information available for deductions than most theorem proving systems have handled in the past. The system has been designed to reduce overhead and to minimize the explosive combinatorics of having many possible deductions to follow. The connection graph is one feature that reduces the number of possible matching expressions.

54. Stickel, M. E., "Theory Resolution: Building in Nonequational Theories," SRI Artificial Intelligence Center Technical Note 286, SRI International, Menlo Park, California (May 1983).

Introduces the concept of separating out nonequational theories (e.g. theories involving inequalities such as those dealing with time) from the normal deduction (resolution) process. By separating these out, special purpose techniques may be brought to bear upon them without hindering the normal deduction. Furthermore, by splitting the proof into two parts, the combinatorics are reduced. The paper presents both total and partial theory resolution.
(To be presented at AAAI-83.)

55. Thorndyke, P., D. McArthur, and S. Cammarata, "AUTOPILOT: A Distributed Planner for Air Fleet Control," Technical Report N-1731-ARPA, The Rand Corporation, Santa Monica, California (July 1981).

56. Trigg, R. H., "A Parallel Approach to the Interactive Design and Simulation of Mechanisms," Technical Report TR-992, Department of Computer Science, University of Maryland, College Park, Maryland (December 1980).

Discusses the distribution of object-oriented simulations within multiple-processor environments.

57. Vere, S., "Planning in Time: Windows and Durations for Activities and Goals," Technical Report , NASA Jet Propulsion Laboratory, Pasadena, California (November 1981).

Describes a planning system that blends concepts of procedural networks, as developed in artificial intelligence, with concepts for scheduling tasks developed in management.

58. Vilain, M. B., "A System for Reasoning about Time," *Proceedings of the National Conference on Artificial Intelligence*, pp. 197-201, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

Describes a set of operators over time intervals and points that maintain relations among them. The system does concern itself with consistency maintenance.

59. Wall, R. S. and E. L. Rissland, "Scenarios as an Aid to Planning," *Proceedings of the National Conference on Artificial Intelligence*, pp. 176-180, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

This system searches a data base of past experiences to find situations similar to the present. These are then modified to fit the present situation and are offered to the user for review. He is able to see , based upon these past experiences, what events are likely to occur in the future.

60. Ward, B. and G. McCalla, "Error Detection and Recovery in a Dynamic Planning Environment," *Proceedings of the National Conference on Artificial Intelligence*, pp. 172-175, Carnegie-Mellon University, Pittsburgh, Pennsylvania (August 1982).

ELMER is a planning system with features for preventing execution errors (by having contingency plans for predictable problems). This paper discusses features for handling unexpected errors. Two approaches to error detection are indicated: (1) explicit error transitions (if event A happens, then an error has been made) and (2) execution monitoring (by which they mean confirmation that what occurs is indeed what was expected). The error recovery methods are specifically associated with the domain and involve retracing of the path taken (by a taxi) or exploring to find the correct path again.

61. Warren, D.H.D., "WARPLAN: A System for Generating Plans," Memo No. 76, Department of Computational Logic, University of Edinburgh (June 1974).

Based upon STRIPS [22], but programmed in PROLOG; uses first-order logic to create plans.

62. Wesson, R. B., "Planning in the World of the Air Traffic Controller," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 473-479, Massachusetts Institute of Technology, Cambridge, Massachusetts (August 1977).

Describes a program that acts as an air traffic controller. The program works by simulating the real world in an idealized world using a production system to generate to respond to events. The program compared favorably to real traffic controllers.

63. Wesson, R., and F. Hayes-Roth, "Dynamic Planning: Searching through Time and Space," Technical Report P-6266, The Rand Corporation, Santa Monica, California (February 1979).

64. Weyhrauch, R. W., "Prolegomena to a Theory of Mechanized Formal Reasoning," *Artificial Intelligence*, Vol. 13, pp. 133-170 (1980).

Describes the ideas behind his FOL (First Order Logic) program. Of particular interest is his simulation structures. A simulation structure provides a computable counterpart to deduction. Rather than using deduction to determine if a fact (e.g. an arithmetic fact) is true, a program that knows about arithmetic can be executed. It may return that the fact is true or false or that it can not decide. This combination of deduction and simulation (execution) is even more important when checking a proposal by deduction may take very much longer than it would take to determine that it was false by simulation.

65. Wilkins, D. E. and A. E. Robinson, "An Interactive Planning System," SRI Artificial Intelligence Center Technical Note 245, SRI International, Menlo Park, California (July 1981).

Discusses a system (SIPE) for planning and plan execution monitoring with plan modification by the user. Uses a procedural network of actions as the representation for the plan. Allows for parallel or sequential actions.

66. Wilkins, D. E., "Parallelism in Planning and Problem Solving: Reasoning about Resources," SRI Artificial Intelligence Center Technical Note 258, SRI International, Menlo Park, California (January 1982).

A further enhancement of SIPE [65]. Reasoning about resources that are used in parallel branches of a plan is streamlined according to the nature of usage of the resource.

67. Wilkins, D. E., "Domain Independent Planning: Representation and Plan Generation," SRI Artificial Intelligence Center Technical Note 266, SRI International, Menlo Park, California (August 1982).

A detailed description of SIPE [65]. This paper describes SIPE's use of constraints, purposes, resources, and logic.

68. Wohl, J. G., "Force Management Decision Requirements for Air Force Tactical Command and Control," *IEEE Trans. on Systems, Man, and Cyber.*, Vol. 11, No. 9, pp. 618-639 (September 1981).

Discusses an interactive technique for the determination and evaluation of options.

MISSION
of
Rome Air Development Center